NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

UTILIZING WEB-BASED TECHNOLOGY TO DESIGN AND IMPLEMENT A CONFERENCE INFORMATION SYSTEM

by

Todd M. Kinney

September 1997

Thesis Advisor:	Monique P. Fargues
Second Reader:	Rex A. Buddenberg

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE **Utilizing Web-Based Technology to Design and Implement a Conference Information System**			5. FUNDING NUMBERS
6. AUTHOR(S) Todd M. Kinney			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER:
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER:
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis is a follow-on effort to work conducted by Michael Chalfant and Kevin Coats [Ref. 1]. The focus is the design and implementation of a web-based information system for the Asilomar Conference on Signals, Systems and Computers. This technical conference specializes in signal and image processing, communications, sensor systems, and computer hardware and software. Organized in collaboration with the Naval Postgraduate School, San Jose State University, and the IEEE Signal Processing Society, the Conference is conducted annually at the Asilomar Conference Facility in Pacific Grove, California. Initial project efforts concentrated on article submissions and system administration (i.e., database management). The article review process and overall implementation of the improved system is the focus of this thesis.

The objectives of this thesis are to: 1) Analyze the article review process of the Asilomar Conference, 2) Implement a World Wide Web (WWW) based article review process, 3) Implement the improved Asilomar Conference information system.

Internet automation is accomplished via interactive WWW pages, created using Borland's Delphi as a programming tool, O'Rielly's WebSite as the web server, and Common Gateway Interface scripts as the mechanism for interactivity. This interactivity provides seamless global access to the Conference database and processes.

14. SUBJECT TERMS World Wide Web, Relational database, Information System			15. NUMBER OF PAGES: 196
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

UTILIZING WEB-BASED TECHNOLOGY TO DESIGN AND IMPLEMENT A CONFERENCE INFORMATION SYSTEM

Todd M. Kinney
Lieutenant Commander, U.S. Navy
B.A. Biology, Wabash College, 1986

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1997

ABSTRACT

This thesis is a follow-on effort to work conducted by Michael Chalfant and Kevin Coats [Ref. 1]. The focus is the design and implementation of a web-based information system for the Asilomar Conference on Signals, Systems and Computers. This technical conference specializes in signal and image processing, communications, sensor systems, and computer hardware and software. Organized in collaboration with the Naval Postgraduate School, San Jose State University, and the IEEE Signal Processing Society, the Conference is conducted annually at the Asilomar Conference Facility in Pacific Grove, California. Initial project efforts concentrated on article submissions and system administration (i.e., database management). The article review process and overall implementation of the improved system is the focus of this thesis.

The objectives of this thesis are to:

1) Analyze the article review process of the Asilomar Conference,

2) Implement a World Wide Web (WWW) based article review process,

3) Implement the improved Asilomar Conference information system.

Internet automation is accomplished via interactive WWW pages, created using Borland's Delphi as a programming tool, O'Rielly's WebSite as the web server, and Common Gateway Interface scripts as the mechanism for interactivity. This interactivity provides seamless global access to the Conference database and processes.

TABLE OF CONTENTS

I. INTRODUCTION

This thesis is a follow-on effort to work conducted by Michael Chalfant and Kevin Coats [Ref. 1]. The focus, both then and now, is the design and implementation of a web-based information system that will automate and provide remote access to the different tasks involved with the organization and administration of the Asilomar Conference on Signals, Systems and Computers. This technical conference specializes in signal and image processing, communications, sensor systems, and computer hardware and software. Organized in collaboration with the Naval Postgraduate School, San Jose State University, and the IEEE Signal Processing Society, the Conference is conducted annually at the Asilomar Conference Facility in Pacific Grove, California.

A. OBJECTIVES

The objectives of this thesis are to expand the design of the information system proposed during the initial thesis effort [Ref. 1], which automates the tasks involved with the Asilomar Conference. To that end, the objectives are to:

1) Analyze the article review process of the Asilomar Conference,
2) Implement a World Wide Web (WWW) based article review process,
3) Implement the entirety of the improved Asilomar Conference information system.

B. OUTLINE

Section II provides background information on the Asilomar Conference, including a general overview of the Conference, Conference organization, and Conference business processes. Section III discusses the efforts of the overall project. Section IV covers the implementation of the project, and focuses on platform and system design issues. Finally, conclusions and recommendations are covered in Section V.

II. BACKGROUND

This chapter provides background information on the Asilomar Conference prior to the start of this project. The first section is a general overview of the Conference itself. The second section details the structural organization of the Conference. Finally, the last section discusses the Conference business processes.

A. CONFERENCE OVERVIEW

The Conference process begins with a "Call for Papers", approximately one year prior to the Conference convening date. This solicitation is designed to generate article submissions under certain predesignated topic areas. Prospective authors submit abstracts and extended summaries of their work. Next, Conference administrators review all submissions in search of those best suited for presentation. Once the selections have been made, acceptance invitations are sent out to the respective authors. Finally, a "Final Program and Abstract Catalog" is printed up and is distributed to potential Conference attendees in order to generate interest and provide a listing of the topics to be presented.

The Conference itself convenes over a three day period in late October. Articles are presented during sessions divided by topic area. Eight parallel sessions occur each half-day, with each session consisting of 8 to 12 presentations. Each Conference attendee is mailed a copy of the Conference proceedings after the Conference.

B. STRUCTURAL ORGANIZATION OF THE CONFERENCE

The Conference is made up of four main elements: organizers, authors, reviewers, and attendees. Organizers handle administrative matters, authors submit and present articles, reviewers select articles for presentation, and attendees attend the Conference. A more detailed discussion of each follows.

1. Organizers

Two organizational bodies, the Conference Committee and the Steering Committee, serve as the guiding forces behind the Conference. The Steering Committee is primarily concerned with long term administration and is not directly involved with the actual setup or execution of the Conference. For this reason, it is not discussed further here. The Conference Committee, on the other hand, is responsible for most of the functions of the Conference. It is comprised of several chairpersons, including the Publicity, Publication, Coordination, and Technical Program Chairs. The Technical Program Chair (TPC) heads the Technical Program Committee, which consists of 6 to 8 Technical Area Chairs (TACs) and is responsible for technical content. The TPC determines what topics should be included in the "Call for Papers", based on previous Conference content and current interest, and administers the article review process. Session Chairs further assist the TPC and TACs by coordinating and scheduling individual sessions. They set the tone for their respective sessions by inviting 3-4 papers covering subject matter that is of particular interest to the session. These "invited" papers are automatically selected for presentation and are exempt from the review process.

2. Authors

Authors that have research pertinent to those topics described in the "Call for Papers" are encouraged to submit their work for consideration. Instructions for article submission are included in the "Call for Papers" and on the Conference Web page. A title, abstract, and extended summary are all that are required to submit an article. Prior to the start of this project, abstracts and plain text extended summaries could be submitted via e-mail or standard mail. After the review and selection process is complete, authors are notified whether or not their articles were selected for presentation via standard mail. Finally, authors that are selected are required to submit a complete copy of their article for use in the publication of the Conference proceedings.

3. Reviewers

The TPC and TACs are responsible for the article review process. The review occurs several months prior to the Conference, the format for which is a massive one day 'face-to-face' review session conducted at the Naval Postgraduate School in Monterey, CA. The Conference sponsors the 'face-to-face' review, an expensive endeavor. During the review session, submitted articles are divided up amongst the various TACs according to topic area, with abstracts serving as the primary means by which articles are judged for acceptance. Extended summaries are also available for review if more details are required. The TPC serves as mediator during the review process in order to resolve any disputes that may occur.

4. Attendees

Information about the upcoming Conference is available to prospective attendees via the Conference WWW homepage or off-line via an established mailing list. Registration occurs manually via standard mail. Bank checks and cash are the only form of payment which the Conference accepts. Following registration, attendees receive an abstract catalog describing the articles scheduled for presentation.

C. CONFERENCE PROCESSES

The Conference Committee is responsible for publishing Conference information. They do so by posting information on the Conference WWW homepage and by sending out information to the mailing list. The Conference Committee collects and sorts registrations and submissions. Following the completion of the review process, the committee sends out accept/reject notices to submission authors and sends out an abstract catalog to prospective Conference attendees. Following the Conference, proceedings are published by the Institute of Electrical & Electronics Engineers (IEEE) Computer Society Press. All abstracts and summaries are discarded two months after the Conference.

III. ASILOMAR CONFERENCE PROJECT

A. INITIAL THESIS WORK

1. Analysis

The first task in the design of the Asilomar information system involved the analysis of the various business functions of the Conference, as the original process needed to be thoroughly understood before improvements could be attempted. Through surveys of authors and interviews with Conference organizers, Coats and Chalfant were able to establish the primary business functions of the Conference: 1) attracting attendees and authors; 2) determining topic areas; 3) collecting article abstracts and summaries; 4) reviewing and selecting articles; and 5) disseminating article information [Ref. 1].

The next task was to determine how these various functions were currently being accomplished, and to what extent automation had already been incorporated into the various functions. The following is a breakdown of the Conference process prior to this project. Existing automated/on-line systems consisted of a Conference WWW homepage where attendees could obtain details about the Conference (location, cost, etc.) and an e-mail address to which abstract submissions could be sent. The non-automated processes included mailing Conference information via a previous-attendees mailing list, collecting article submissions through standard mail, determining topic areas, reviewing articles, publishing the hard-copy Conference proceedings, and presenting author papers [Ref. 1].

2. Planning

Analysis of the overall process suggested that automation of several of the Conference's key business functions would decrease expenses and increase efficiency. In particular, both the submission and review processes are time consuming and expensive, as authors have to mail articles via either hard copy or electronic format via e-mail. Conference administrators have to collect the information, enter it into a database, collate the abstracts and summaries, and arrange and print the Abstract Catalog. The review process is even more inefficient and expensive. The Conference incurs significant expense in the transport and lodging of reviewers during the one-day marathon review session that occurs annually at the Naval Postgraduate School in Monterey, CA. Although the review process seemed more in need of re-engineering, logic dictated that it was first necessary to redesign the submission process. For this reason, and due to time constraints, Coats and Chalfant decided that the scope of their endeavor would necessarily have to be limited to the submission portion of the system, along with administrative functions.

3. Design

The submission process and several administrative functions were automated through the use of HTML 3.0 and Common Gateway Interface (CGI). The automation allows authors to submit information about their articles via the WWW and any common browser. It also allows for the update of their contact information maintained in the Conference database.

Several system administration functions were also automated, such as maintaining passwords, accessing the database, updating the database as articles are reviewed and selected, and printing articles, accept/reject notices, and the abstract catalog [Ref. 1]. User-friendly interfaces and highly automated CGIs were used, keeping 'ease of use' in mind.

4. Implementation

The automation of the submission process and the administration process was completed by Coats and Chalfant in September of 1996. However, the new prototype was used only sparingly during the 1996 Asilomar Conference. Authors submitted their article abstracts and extended summaries via standard mail or electronic mail. The administrative staff collected the submission data and manually transferred it into the prototype system database using an ordinary browser and the WWW. This served as a way to further debug the portions of the system designed and implemented by Coats and

Chalfant [Ref. 1]. Several administrative tasks, such as the setup of the abstract catalog and the advance program and generation of acceptance/rejection notices, were then conducted via the System Administration Functions of the new prototype system. The goal of both thesis efforts is to have full functionality and implementation of the new prototype for the 1997 Asilomar Conference.

B. FOLLOW-ON THESIS WORK

1. System Assessment

As mentioned in the previous section, the system automation conducted by Chalfant and Coats dealt primarily with the On-line submission of articles and some system administration functions. This was a necessary precursor to the real benefits to be gained from automation. The article review process constitutes a significant expense for the Conference, as various costs to fund the review session in Monterey are sponsored by the Conference organization. Cost reduction alone would be reason enough to modify the review process to allow for remote access reviewing. In addition, the amount of time available for reviewers during the review session is insufficient for providing a thorough review for each article submission. Remote access reviewing would allow reviewers more time to conduct individual reviews. Reviews could be conducted at any time from any location with an ordinary browser, as long as WWW access is available. The result would be a more thorough, fair and impartial review process.

2. On-line Article Review Sub-system

The On-line Article Review Sub-system makes use of Common Gateway Interface (CGI) programs and HTML forms in order to interactively access article submissions. Reviewers can peruse the various article submissions via keyword searches of the database. Once they have reviewed the abstract, they can submit their vote and any comments they may have about the submission via the HTML form. The review data is stored in the Reviews database table, where it can be accessed later by the reviewer for editing purposes or by the Master Reviewer for accounting purposes.

3. Master Review Sub-system

The Master Review Sub-system, also employing CGI programs and HTML forms, allows the Master reviewer to access the Reviews database table in order to assess the submitted reviews. The Master reviewer can view the vote tally for each article and read the various reviewer comments. The system also allows the Master reviewer to initiate accept/reject action based on votes submitted by the reviewers. Once the decision is made, the push of a button updates the accept/reject information in the system administration database. In the event of a tie, the Master reviewer has final decision authority.

4. Security

There are a number of security concerns involved with the utilization of a Web-based Conference information system. Secrecy and integrity of both article submissions and reviewer inputs are crucial for the fair and impartial conduct of any Conference process. Secrecy is an issue because reviewers require the ability to vote freely, without fear of repercussion from other reviewers. Review submissions must therefore be kept confidential, their access restricted from anyone other than the originating reviewer and the Master Reviewer. The system accomplishes this via password protection. Only the Master reviewer has access to all of the reviewer inputs.

Integrity is an issue because of the copyrights involved with article submissions. Article data is, theoretically, copyrighted the moment it is placed into a "tangible medium", such as a text, .pdf, or .ps file format [Ref. 10]. Unauthorized tampering with article submissions is, therefore, illegal and unethical. The system addresses data integrity via both password protection and user access restrictions (see section IV.A.2).

The security discussion thus far has been limited to secrecy and integrity as described above. Although it is not germane to the Asilomar Conference on Signals, Systems and Computers, it should be noted that secrecy, when dealing with classified information, requires additional security measures, such as data encryption and decryption.

5. System Upgrades

Several system application upgrades were suggested following the initial project and are, subsequently, implemented during this thesis. First, the Web server software, O'Reilly's WEBSITE, Version 1.0 [Ref. 8], has been upgraded to WEBSITE Professional, Version 1.0 [Ref. 8]. The Professional version provides support for more Application Programming Interfaces (APIs) and provides more security features, such as Secure Socket Interface for secure data exchange.

Additionally, the FTP server software, WFTPD Version 1.1, was upgraded from the 16 bit version to the 32 bit version, WFTPD Version 2.34 [Ref. 7]. This upgrade provides improved performance as well as an improved user interface.

6. System Migration

Once all of the application upgrades were in place, all programs for the various sub-systems complete, and all security measures tested and implemented, the next step was to migrate the Conference and its processes to the new system. Placing the system 'on-line' was simply a matter of activating the link to the On-line Article Submission sub-system located on the Asilomar homepage.

IV. IMPLEMENTATION

A. ISSUES

1. Applications

a. Delphi

Borland's Delphi [Ref. 9] is one of the most powerful and easy to use Rapid Application Development (RAD) tools available on the market today. It is truly object oriented, using Object Pascal as its programming language. Some of its attractive attributes are its easy-to-use "drag-and-drop" interface and its scalable database connectivity. More importantly, Delphi is a programming tool that makes for the simplified creation of Common Gateway Interface (CGI) programs. These CGI programs, or scripts, are what makes dynamic Web pages possible. CGI and dynamic interactivity are discussed in further detail in the next section. For now, suffice it to say that Delphi serves as a vital link between client and server in the retrieval and dissemination of information. For all of these reasons, Delphi was chosen as the application development tool for use by this project.

b. Common Gateway Interface

The tremendous growth and popularity of the WWW is largely attributable to CGI. This standardized protocol is the vehicle by which Web use has evolved from the viewing of static Web pages to user interaction with dynamic Web documents. Dynamic or 'live' data has opened up a world of possibilities for the WWW and its users.

For our purposes, CGI is the mechanism or agent by which Conference functions can be automated. Whereas article data was once submitted via e-mail or standard mail and entered manually into the database, now information can be submitted interactively via the Web with any standard browser. More importantly, for administrators, the submission data is automatically entered into the database at the time of browser submission. A narrative describing how CGI accomplishes this task follows in the next paragraph.

CGI is a process or specification that resides external to the Web server software. CGI scripts for this project were created using Delphi and are contained within the Delphi compiled executables. When a Web site receives a call from an HTML form (i.e., a user request) specifying a certain CGI script, the Web server forwards input data from the requesting browser to the CGI script via a temporary file that it creates solely for the execution of the CGI script. Upon receiving the input data, the CGI script executes the requested task. CGI scripts, which can perform any number of tasks, provide functionality for Web servers. Database access and administration is the primary functionality that CGI brings to this project. The process is relatively simple, and is depicted graphically in Figure 4-1. A user accesses the Asilomar Web site using an ordinary browser (e.g., Netscape Navigator or Microsoft Explorer). Entering data into a form at this Web site triggers the WEBSITE web server to execute a CGI script.

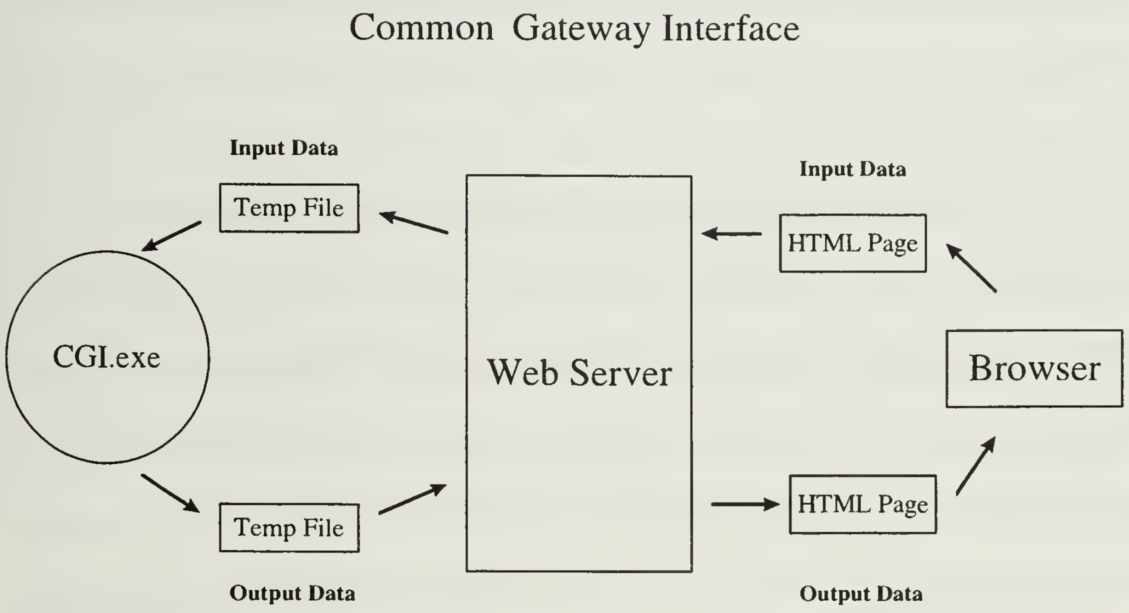


Figure 4-1

WEBSITE calls the executable file and passes the CGI input parameters from the form via a temporary file. The CGI executable reads the temporary file, processes the data and sends the output data stream back to the temporary file. When WEBSITE detects that the CGI script has ceased (it finds the expected output file returned by the CGI code), it retrieves the data from the executable and then sends the data to the browser [Ref. 3].

The important thing to note about the CGI process is that the executable file exists and operates external to the Web Server software. In fact, the CGI executable shares CPU resources with the Web Server. This provides flexibility in what CGI programs can do. The drawback, however, is the increase in resource overhead required by the CGI executables.

CGI is by far the most prevalent method of interactive Web page implementation in use today. There are several reasons for this. First, CGI is flexible and adaptable to users' needs. Second, it is a standardized and established protocol. Third, developmental costs and risks are low compared to most other gateway interfaces. The primary drawback of CGI is poor performance under certain conditions, such as concurrent page hits. When multiple users simultaneously request the same CGI process, multiple resource intensive CGI programs start running concurrently, each contributing significantly to system overhead. The result is system slowdown in the form of poor performance. In fact, the ability or tendency of CGI programs to monopolize system resources necessitates the utilization of a true multitasking operating system, such as Windows95 or WindowsNT. However, in an environment where multiple concurrent hits are not expected (i.e., only one CGI program executes at any given time), required system overhead will be minimal and poor performance will be a nonissue [Ref. 6].

One alternative to CGI that has gained some notoriety is Application Programming Interface (API). API, like CGI, is a protocol that provides increased functionality to a Web Server. Unlike CGI, API achieves this added functionality through the use of a dynamic link library (DLL), not through the use of external programs. APIs, existing as DLLs, reside as internal processes within the Web Server software. They function similarly to CGI programs, called into execution by user browsers via HTML forms. The main difference between CGI and API is that API DLLs run as internal processes within the Web Server memory space. Improved performance of memory executed API versus hard disk executed CGI can be significant, but not enough to offset the drawbacks encountered in the use of APIs. First, API is not a standardized protocol like CGI. It is proprietary, normally tied to the manufacturer of a particular Web Server. For instance, two popular API variants are Netscape's NSAPI [Ref. 11] and Microsoft's ISAPI [Ref. 12]. Second, the development of API programs is an expensive and time consuming process. API, unlike CGI, involves a lower level of programming, which requires knowledge and understanding of such system intricacies as multithreading and multitasking. This requires a degree of programming expertise not often found 'in-house'. External agents or consultants are often hired to develop these applications, making the System Development Life Cycle of API programs costly and time consuming [Ref. 6]. In addition, the lack of familiarity with the organization often makes the external agent's task that much more difficult. CGI programs can usually be developed 'in-house' by organizational personnel knowledgeable of both programming and of the organization itself. Third, API programs are a risk because they execute within the Web Server's address space as DLLs. The failure of a corrupt API program can easily bring down the entire server. A corrupt CGI program, on the other hand, is not likely to crash the server. Finally, updating and modifying CGI forms is far easier than modifying API forms. This is important, considering how often pages are likely to require change and/or modification.

These considerations led to the choice of CGI as the Web Server protocol for this project, CGI provides both functionality and flexibility at an affordable cost. Additionally, the scenario under which this prototype will be utilized is not one subject to multiple concurrent hits that might result in poor CGI performance. The system does involve many CGI programs, but the likelihood of more than one being called upon at any given time is low. Finally, long term maintenance was also a consideration. Should a form or page changes be required, CGI coding and debugging is far more manageable than complex API programs.

c. *Web Server*

The Web server software serves as the controlling agent between the Web site (the server) and the Internet user (the client), maintaining the connection between the PC hardware and the Internet. The PC used in this project has been assigned a static IP address and maintains a continuous Internet connection via the ECE department's local area network. When an Internet user accesses the PC's address, WEBSITE automatically displays **Index.htm** in the C:\Website\Htdocs\Submit directory. WEBSITE then displays other HTML pages or executes Delphi programs as implemented within the system.

All HTML pages are stored according to their sub-system category. On-line Article Submission pages are stored in the C:\Website\Htdocs\Submit directory. On-line Article Review and Master Review pages are stored in the C:\Website\Htdocs\Review directory. Finally, System Administration pages are stored in the C:\Website\Htdocs\Admin directory. All of the Delphi executables for all of the sub-systems are stored in the C:\Website\cgi-win directory as required by WEBSITE. The database queries conducted throughout the system are generated via the Delphi executables, with WEBSITE serving as the intermediary between the browser (user) and the system. WEBSITE passes the query via standard query language (SQL) to the system database, Borland's PARADOX.

One of the primary responsibilities of the WEBSITE Professional web server software is security. WEBSITE accomplishes this through the judicious control of user access and rights to the web server and its directories. WEBSITE limits a user's access to the C:\Website\Htdocs directory and above, and strictly prohibits access to the root directory. WEBSITE also possesses FTP download capability. This functionality allows users to download any file with an '.exe' or '.html' extension that resides in the C:\Website\Htdocs directory or higher. FTP uploads, on the other hand, are handled strictly by the FTPD FTP server, which is discussed in the following section.

d. FTPD FTP Server

The system requires that authors be allowed to upload their Extended Summaries to the Conference database. Because summaries may contain graphics and equations not suitable for ASCII text, they cannot be submitted in the same manner that the article abstracts are, via HTML forms. They must be submitted as separate files in a separate manner. FTP is an Internet utility that provides the file transfer functionality necessary for the varied file formats encountered in Extended Summaries. WFTPD, Version 2.34 (32 bit version for Windows 95) [Ref. 7], was chosen as the FTP server for the project based on its balance between affordability and functionality.

Allowing uploads presents a security risk to the server due to the potentially dangerous types of files that could be introduced to the system. System administrators are warned not to execute any unknown programs (.exe files) found in the upload directory. Such programs should be deleted immediately upon discovery. The only legitimate file formats for uploading Extended Summaries are postscript (.ps), .pdf, or plain text (ASCII). FTP security is discussed in further detail in section IV.2.c.(2).

e. Database

The relational database used in the Asilomar Conference process is Borland's PARADOX [Ref. 9]. Database implementation for the system during the initial thesis mirrored the one already in use by Conference administrators. Seven tables were constructed to maintain data on the Contact Authors, Submissions, Sessions, Keywords, and system passwords. Three additional tables were constructed during this portion of the project. The first two are *UserID* and *Password* tables for the reviewers and the Master Reviewer. The other is a Reviews table for maintaining article review information.

1) **Review.DB** contains reviewer ID's and Passwords.
2) **Master.DB** contains the Master Reviewer ID and Password.
3) **Reviews.DB** contains the article review information submitted by the various reviewers.

These tables are depicted in Appendix B.

2. Security

Security is an important concern for the Conference organization. Authors from all over the world may submit article abstracts and extended summaries as well as mailing information via this system. Conference reviewers will utilize the system to conduct 'on-line' reviews in order to determine which articles should be presented during the actual Conference. Finally, Conference administrators will use both the author and article information to print the abstract catalogs and the Conference proceedings. For all of these reasons, the integrity and security of Conference submission data become paramount.

a. System Protection

The physical security for this prototype is provided primarily by the physical location of the server itself, which is isolated from routine traffic. In addition, the server is password protected on boot-up and uses a password protected screen saver. This setup obviously does not provide complete access restriction, but the nature of the Naval Postgraduate School environment and the integrity of the student body and faculty does not warrant stricter measures. The real threat to the server is from the Internet, not from physical tampering.

b. Database Protection

The Conference database is protected by conducting regular backups and by restricting access to the database. Full magnetic tape backups of the entire database, conducted in a timely fashion on a regular basis, provides insurance against catastrophic loss of data. Physical access to the database, which resides on the server, is limited as described above, using a remote location and a password protected screen saver. On-line access to the system via the Internet poses a greater security threat, but is addressed in the design of the system. Database access and functionality throughout the system is governed by password protection.

One problem with Web site management and Internet applications is that all executable and HTML files located on the Web server (C:\Website\Htdocs and below) are available to anyone accessing the site for download or execution. This means that any of the system's CGI scripts can be executed via an ordinary browser if the URL path of that particular program is known.

Fortunately, the stateless nature of CGI and HTML lessens the severity of this particular security risk. Because all of the CGI scripts are dynamic, the necessary input parameters must first be passed as hidden input from the forms directly preceding the database executables. Any would-be hacker entering the path of the CGI script, minus the input parameters, would receive only an empty form.

Additionally, this security risk is addressed in a more proactive manner by programming a conditional check of a flag variable into every executable. This flag can only be set by entering the appropriate login and password combination on the login screen. Once the password has been entered correctly, a flag is set that will allow a user to access all database functions. If the flag is incorrect (i.e., incorrect password), the program will not execute and a warning message will be displayed along with a hyperlink back to the login screen. So, even if an unauthorized user discovers the name of an executable, the flag variable still requires the proper login and password [Ref. 1].

Unfortunately, as secure as this setup may appear, the threat of a brute force attack on the login and password combinations of the system still exists. This type of attack is one where hackers employ special software designed to guess the UserIDs and passwords of legitimate users. The best safeguard against this type of threat is administrative in nature. The organization must train system users on proper password selection and utilization techniques [Ref. 1]. It should also be noted that no matter how secure the end system may be, electronic submissions are still vulnerable to eavesdropping while enroute, unless some sort of encryption scheme is used.

c. Application Protection

(1) WEBSITE Security. The denial of access to the PC's hard drive is the responsibility of the server software, WEBSITE. This program controls all Internet access functions of the system. Inherent in its design is a restriction that limits users to directories higher than the C:\WebSite directory. In other words, when a browser accesses the system, the lowest directory available to it is either C:\Website\Htdocs or C:\Website\cgi-win. It is not possible to access the root directory or any other directory than those stated above. WEBSITE also does not allow file uploads (i.e., a user sending a file **to** our computer) or modification of files on our system. Therefore, a user may look at and execute any file that we place higher in the directory structure than C:\WebSite, but nothing else.

WEBSITE provides several options for managing access control to the Web server. The WEBSITE server administration control panel allows the system administrator to restrict access to various directories based on user, group, and/or IP address. For example, the directory C:\Website\Htdocs\Personnel could be established for the Personnel Department. The system administrator wants to limit access to this directory (and the Web pages contained therein) to only those employees of the Personnel Department. This could be accomplished by establishing a group entitled, "Personnel", and restricting access to that directory to only members of that group. The problem here is that every member of the Personnel Department must be assigned a *UserID* and then added to the Personnel group, contributing significantly to the workload of an administrator when the number of departmental employees is large.

Another option is to restrict access to the directory based on IP address. Only users connecting from authorized IP addresses are granted access to the directory. This works well if the number of IP addresses is small and access to them can be controlled. When the number of users or employees is large, this approach eliminates the administrative overhead associated with assigning numerous *UserID*'s.

Finally, a combination of user, group, and IP address restrictions can be implemented by the system administrator. This is the approach that we have taken with the System Administration, On-line Article Review, and Master Review Sub-systems. Whereas the Article Submission Sub-system has no access restriction associated with its use, these other sub-systems require a greater degree of secrecy and integrity. For this reason, the 'admin' and 'review' groups were created for restricting access to their respective sub-systems. A 'Master Reviewer' user was created for access to that sub-system. In addition, the IP addresses associated with key Conference personnel have been added to the access control list. WEBSITE utilizes 'and/or' logic when determining user access. When WEBSITE receives a call or hit from the WWW, it performs an access check based on the particular URL address that is being called upon. WEBSITE first checks to see if the calling IP address is on the access list. If it is not, then WEBSITE prompts the user for a *UserID* and *password.* If the user is on the access list and the password is correct, access is granted.

(2) FTPD Security. If not configured correctly, an FTP server can be a security risk. Security must be inforced through the judicious use of user access rights built into the FTP server software package. User's with unlimited access rights can not only view what directories and files are on the server, but they can write to the directories and modify files as well. Malicious user's could easily damage important server data or even wipe out the contents of the entire server. For this reason, both reviewers and authors have been granted limited FTP access rights. The current configuration restricts authors to the upload directory only and limits them to 'write-only' access. This means that authors can write their extended summaries to the upload directory, but they cannot view the contents of the directory, modify its contents, or change directories. By limiting the user's ability to change directories, modify directory contents, and view directory contents, the ability to corrupt another author's data is slight. Reviewers are granted 'read only' access to the upload directory. This allows them to read the various extended summaries, but does not allow them to modify the contents of files in any way.

B. SYSTEM DESIGN

1. System Overview

The Asilomar Conference information system is composed of five major sub-systems:

1) The Asilomar Conference Information Homepage,
2) The Article Submission sub-system,
3) The System Administration sub-system,
4) The Article Review sub-system,
5) The Master Review sub-system.

The first sub-system, the Information Homepage, was in existence prior to this project. The Article Submission and System Administration sub-systems were developed and implemented, for the most part, by Coats and Chalfant [Ref. 1]. The Article Review and Master Review sub-systems, as well as a few System Administration programs, are developed and implemented in this follow-on thesis. Images of each HTML page are at the end of this chapter and are referred to in the program descriptions below. All of the code for the programs and forms (HTML, Delphi, and CGI) are included in the appendices for each of the various sub-systems.

2. On-Line Article Review Sub-system description

The On-Line Article Review sub-system is a combination of HTML pages and Delphi executable programs that form the user interface for the Conference article review process. This sub-system allows remote access to article submissions so that program chairs can review article submissions from anywhere in the world via the WWW using an ordinary browser. The sub-system is password protected in order to insure only authorized Conference reviewers access the article submission database.

a. Log-on page

The On-line Article Review Log-on page (**Review30.htm**) serves as an authentication and access control mechanism for the system. In order to access this page (Figure 4-2), a reviewer must have already passed the first level of access control presented by WEBSITE. This form solicits both the *ReviewerID* and *Password* from the reviewer. When this data is entered, the browser signals the web server to execute the underlying CGI script, which verifies the *ReviewerID* and *Password* with the reviewer password database located in the **Review.DB** table in PARADOX on the server. If the ID and password match, then the reviewer is presented with the main menu page of the On-line Article Review System.

b. *Main Menu page*

The On-line Article Review Main Menu page (**Revpwd.exe**) welcomes the reviewer to the On-line Article Review System and presents the available review functions: 1) *Review an Article*, or 2) *Edit a Review*, as shown in Figure 4-3.

c. *Review an Article*

The *Review an Article* (**Artrev.exe**) function generates a page that presents three different methods by which a reviewer can search the article submission database for articles to review, as shown in Figure 4-4. The first is an 'All Article' search that lists all of the article submissions currently in the Conference database. The second search method is a keyword search that allows reviewers to select from a list of keywords in a drop-down option box. The third search method is a Paper Number search that allows reviewers to query the database for a specific Paper Number. The first two search methods initiate the **ArtSrch2.exe** CGI program, which executes the appropriate query of the Conference database. The search results are presented in table format on the **ArtSrch2.exe** page, listing Paper Number, Title and Contact Author for each submission, as shown in Figure 4-5. If the article of interest is found amongst the titles listed, it may be viewed by selecting the appropriate radio button and pressing the 'submit' button at the bottom of the table. This will prompt the system to execute the *Article Review page* (**ArtView2.exe**) CGI script, which displays all of the article data and provides reviewer input tools, as shown in Figure 4-6. The Paper Number search differs from the first two search methods in that it bypasses the **ArtSrch2.exe** CGI program and takes the reviewer directly to the **ArtView2.exe** page, since the specific Paper Number is already known and provided by the reviewer.

The **ArtView2.exe** page is the heart of the On-line Article Review sub-system. It allows reviewers to remotely access articles in the database and read them at their convenience. Here is where all vital article information (paper number, title, authors, and abstract) is displayed for the reviewer's perusal. Additionally, a hyperlink to the upload directory is provided that allows the reviewer access to the article's extended summary.

Review input utilities are located on the form just below the article data. Radio Buttons provide a mechanism for reviewers to vote on whether or not the article should be accepted for presentation at the Conference. Five vote categories are available: 1) Strong Accept, 2) Conditional Accept, 3) Reject, 4) Undecided, and 5) Not Qualified. With the exception of 'Conditional Accept', most of these are self-explanatory. 'Conditional Accept' is used to indicate that the reviewer favors the submission on the condition that sufficient room is available for its inclusion without conflicting with more worthy candidates. Stronger papers take precedence. A text area, provided below the 'Vote' section, allows for reviewer comments. Here, reviewers can add amplifying remarks about the article and their vote. For example, reviewers voting ‘Conditional Accept’ could provide amplifying remarks with regard to the strength of their commitment to the condtionality of the accept.

Once the review is complete, pressing the 'Submit' button prompts the system to add the review information to the **Reviews.DB** table. This is accomplished via the **Add133.exe** CGI program. A 'Thank You' page is generated, providing confirmation to the reviewers that their input has been submitted, as shown in Figure 4-7. The reviewer then has the option of either conducting another article review or returning to the Review System Main Menu.

d. Edit a Review

The *Edit a Review* (**EditRev.exe**) function generates a page with a table containing all of the reviews in the **Reviews.DB** table by that particular reviewer (Figure 4-8). The CGI program accomplishes this by conducting a query of the **Reviews.DB** table using the *ReviewerID*, which is passed from the initial log-on page as hidden input. The format for this table of reviews is the same as that described in the *Review an Article* section, listing Paper Number, Title, and Author. The reviewer selects the review of interest in the same manner as before, clicking on the appropriate radio button and pressing submit.

The system executes the *Review a Review* (**Revview.exe**) CGI script, which produces a page containing both article data and previous reviewer input data. Once again, the reviewer can read the article abstract and Extended Summary and, if necessary, modify his or her vote and/or comments from the previous review, as shown in Figure 4-9.

Once the edited review is complete, pressing the 'Submit' button prompts the system to update the review information in the **Reviews.DB** table. This is accomplished via the **Add133b.exe** CGI program. A 'Thank You' page is generated, providing confirmation to the reviewer that his or her input has been submitted, as shown in Figure 4-10. The reviewer then has the option of either editing another review or returning to the On-line Article Review Sub-system Main Menu.

3. Master Review Sub-system description

a. Log-on page

The Master Review Sub-system Log-on page (**Masterev.htm**) serves as an authentication and access control mechanism for the system. In order to access this page (Figure 4-11), the Master Reviewer (usually the Technical Program chair, who has final authority regarding acceptance and rejection of submissions) must pass the first level of access control presented by WEBSITE (UserID and Password). The form itself initiates a second level of access control by soliciting both the *MasterID* and *Password* from the Master Reviewer. Once this data is entered, the browser signals the Web server to execute the underlying CGI script, which verifies the *MasterID* and *Password* with the password database (**Master.DB**) located in PARADOX. If the ID and password match, the Master Reviewer is presented with the Main Menu page of the Master Review System.

b. Main Menu page

The Master Review Sub-system Main Menu page (**Revpwd.exe**) welcomes the Master Reviewer to the Master Review Sub-system and presents the available Master Review functions: 1) *Summary of Reviews*, 2) *Accept/Reject Articles*, or 3) *Overall Submission Status*, as shown in Figure 4-12.

c. Summary of Reviews

The *Summary of Reviews* (**Revsum.exe**) function generates a page that presents three different methods by which the Master Reviewer can search the article review database (Figure 4-13). The first method displays a table that lists all reviews in the database. The second method is a keyword search. The Master Reviewer selects from a list of article keywords in a drop-down option box. Once the selection is made, whether it is an 'All' search or a 'keyword' search, the **Revsrch.exe** CGI program executes the appropriate query of the database and returns the resultant reviews in table format, as shown in Figure 4-14. Finally, the Master Reviewer may also search the **Reviews.DB** table by *ReviewerID*. A table is returned listing all reviews by that particular reviewer.

As described in the On-line Article Review section, once the Master Reviewer finds the title for which he or she wants to the summary of reviews for, he or she simply selects the article of interest and clicks on the 'Submit' button. The associated CGI program (**Revsrch2.exe**) returns all of the article information, plus a table that summarizes the reviews that have been submitted by the various reviewers (Figure 4-15). The votes and comments of each reviewer is listed in the first table. A table that tallies the vote total is displayed next.

Based on this information, the Master Reviewer can decide whether or not to Accept or Reject the article for presentation at the Conference. Accept/Reject buttons are provided at the bottom of the form to allow the Master Reviewer to accomplish this very task. Pressing either of these buttons executes the **MastAcep.exe** CGI program, which updates the Accept/Reject field of the **Submissi.DB** table for the article in question. A confirmation page is returned, providing verification that the database has been modified accordingly, as shown in Figure 4-16. The Master Reviewer then has the option of either viewing another summary or returning to the Master Review System Main Menu.

d. Accept / Reject Articles

This function is identical to the Accept/Reject Articles function found in the System Administration sub-system. It allows the Master Reviewer to go in and directly Accept or Reject any article in the submission database. Changes may be made to one article, to several articles, or to all of the articles.

1) Select Articles (**Acceprej.exe**)

2) Accept / Reject Articles (**Aceprej2.exe**)

(1) Select Articles (**Acceprej.exe**). This program displays a form with a list of all the article titles in the **Submissi.DB** table, each with a checkbox (Figure 4-17). A drop-down option box allows the administrator to either "accept," "accept then reject," "reject," or "reject then accept" the selected titles. These multiple choices simplify the Accept/Reject process in that the Master Reviewer or system administrator does not have to individually mark every single article. When a list exists with more rejections than acceptances, checking off accepted articles and the "accept then reject" option will set the selected titles as accepted and automatically set the rest as rejected. If there are fewer rejected articles than accepted, then the rejected article titles are selected and "reject then accept" will reject those and accept the rest [Ref. 1].

To mark articles individually, either the "accept" or "reject" option is selected on the drop-down box and the title(s) checked off are modified as appropriate. "Accept" or "reject" will only modify the selected articles and will not affect other articles. Submitting this form calls **Aceprej2.exe**, as shown in Figure 4-18.

(2) Accept/Reject Articles (**Aceprej2.exe**). The call to this program carries a query string containing those articles marked in the previous form.

Aceprej2.exe receives and parses the query string to obtain the action and the titles. A case statement uses the action passed to modify the necessary article fields in the **Submissi.DB** table. Titles are placed on a string list and individually processed. When the action is set to "accept then reject" or "reject then accept," all articles are first set globally to the second action (either "reject" or "accept", respectively) and then the string list is processed to change the selected articles to the first action (either "accept" or "reject", respectively) [Ref. 1].

e. Overall Submission Status

The *Overall Submission Status* program (**Overstat.exe**), when called, executes a query of the **Submissi.DB** and **Reviews.DB** tables and returns a table listing the following information: 1) total number of article submissions, 2) total number of articles reviewed, 3) total number of articles accepted, and 4) total number of articles invited, illustrated in Figure 4-19. This provides the Master Reviewer with a 'big picture' of where the overall Conference process stands.

4. Miscellaneous System Admin Programs

a. Overall Submission Status

This function (**Overstat2.exe**) is identical to that previously described in the Master Review sub-system (**Overstat.exe**). It is duplicated in the System

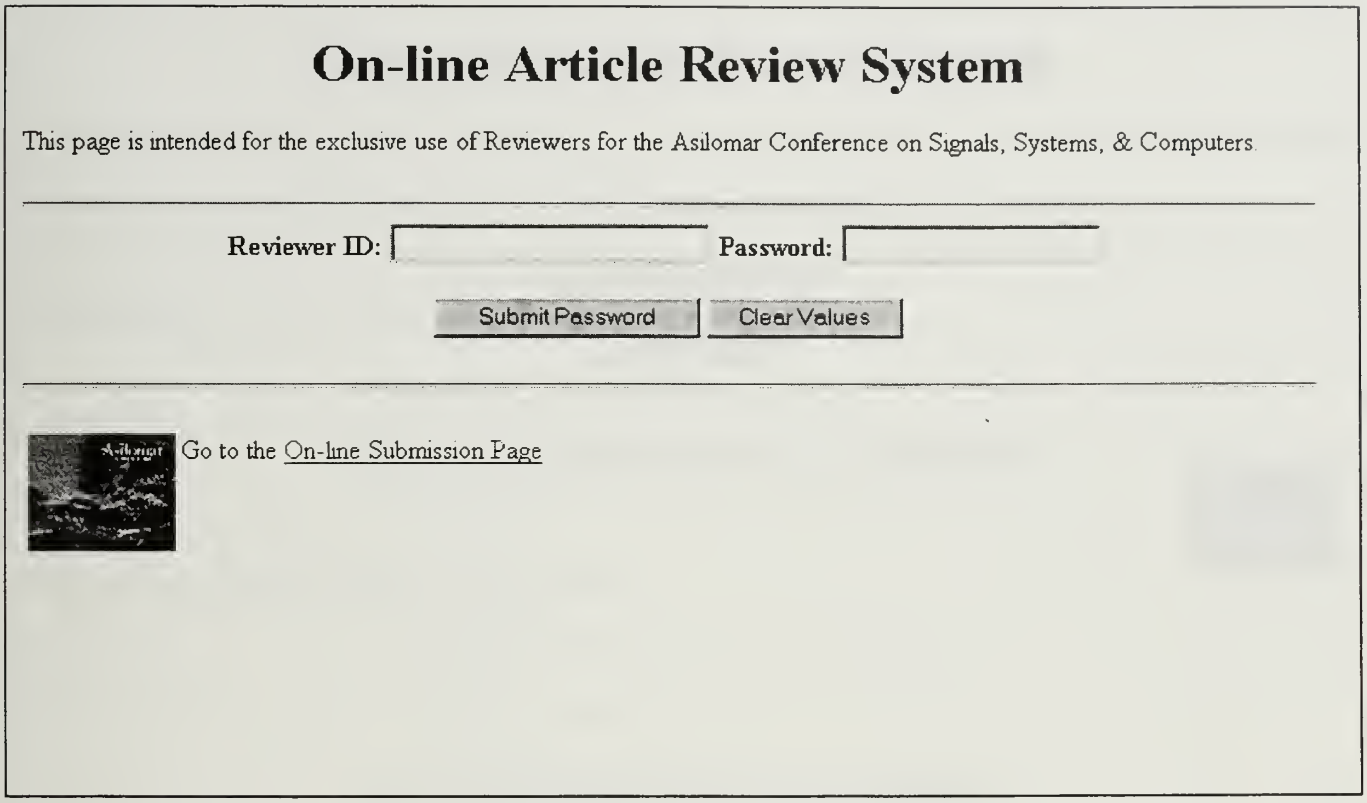
Administration sub-system simply for administrative convenience, as shown in Figure 4-20.

b. Database Purge

This function was added to the system to allow administrators to purge the submission and review databases each year following the Conference. This frees up storage space and avoids confusion when the following year's submissions start to roll in. Once selected, the purge program (**Timeout.exe**) displays a confirmation page that requests verification from the administrator that he or she 'really' wants to purge the database of old entries, as shown in Figure 4-21. If the user selects 'continue', the **Timeout2.exe** program executes a CGI script that purges both databases based on a search of the date of each entry. All records with date/time stamp older than 15 days are purged from both the **Submissi.DB** and the **Reviews.DB** tables. A 'Year' field (that incorporates the server date/time stamp at the time of record entry) was added onto the **Submissi.DB** table and incorporated into the design of the **Reviews.DB** table in order to add this capability. Whenever an article submission or review are added to one of their respective database tables, the current Date/Time stamp is acquired from the Server PC's bios and is added to the database record entry under the Year field. A confirmation page is posted following the purge, informing the administrator of exactly how many records were purged from each of the databases (Figure 4-22). The administrator is then presented with a hyperlink back to the System Admin Main Menu page.

5. System Web Pages

This section contains screen captures of the Web pages generated by the various Asilomar Conference sub-systems developed during this thesis. The pages are viewed with Netscape Navigator Gold 2.01 [Ref. 13] and are captured with HyperSnap-DX, Version 3.02 [Ref. 14].



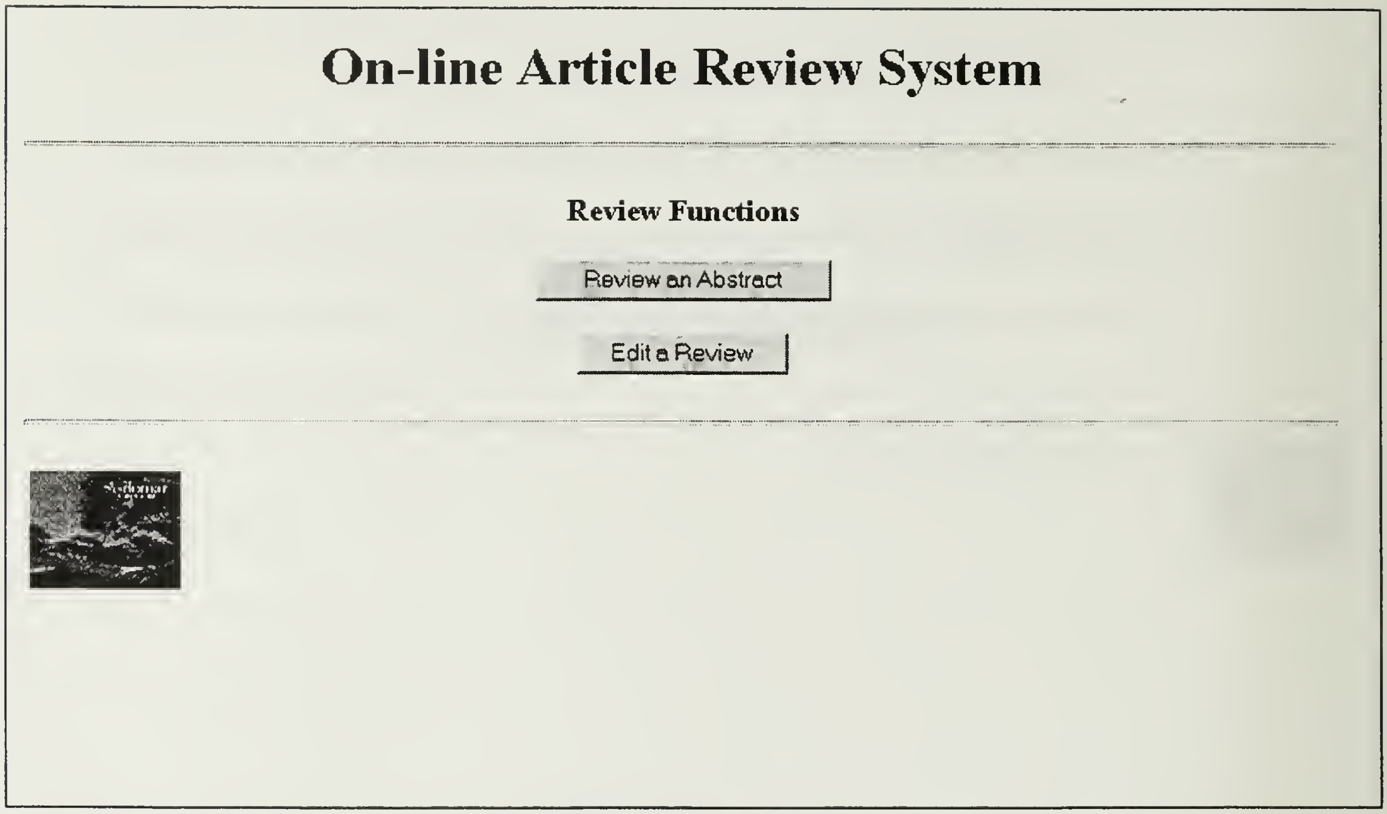
On-line Article Review System

This page is intended for the exclusive use of Reviewers for the Asilomar Conference on Signals, Systems, & Computers.

Reviewer ID: Password:

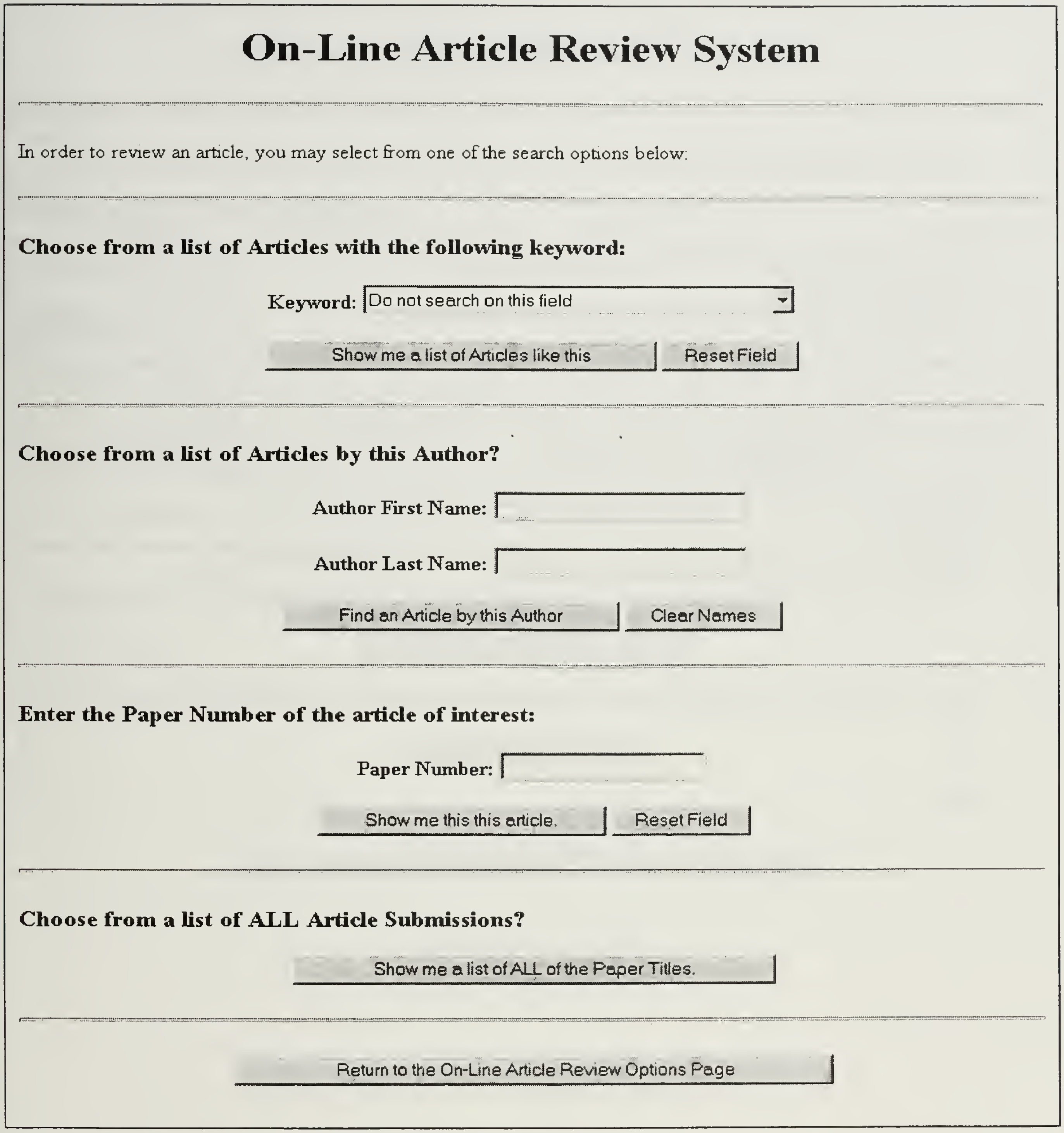
Go to the On-line Submission Page

Figure 4-2. Review30.htm.


On-line Article Review System

Review Functions

Figure 4-3. Revpwd.exe.


On-Line Article Review System

In order to review an article, you may select from one of the search options below:

Choose from a list of Articles with the following keyword:

Keyword: Do not search on this field

Show me a list of Articles like this
Reset Field

Choose from a list of Articles by this Author?

Author First Name:

Author Last Name:

Find an Article by this Author
Clear Names

Enter the Paper Number of the article of interest:

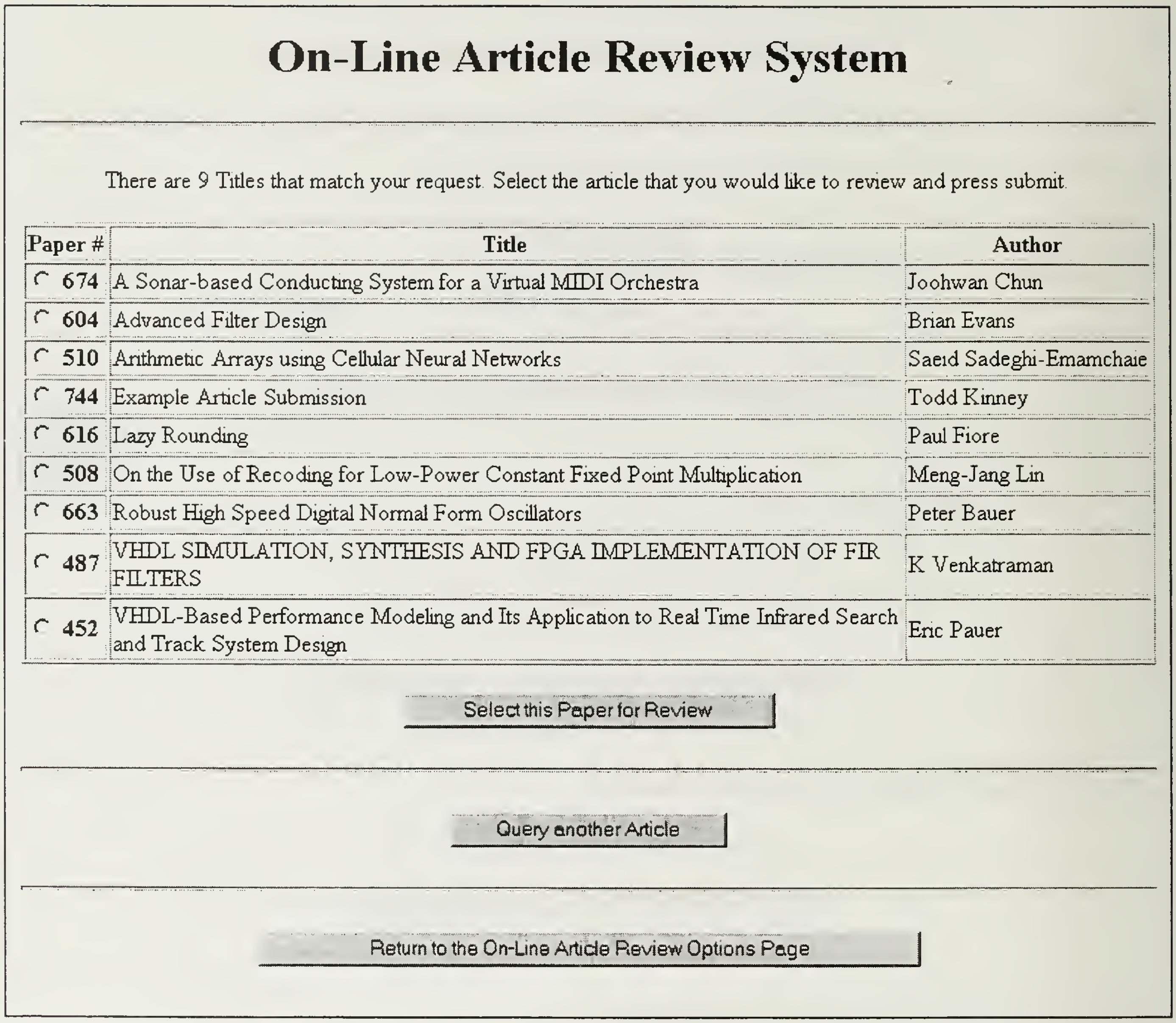
Paper Number:

Show me this this article.
Reset Field

Choose from a list of ALL Article Submissions?

Show me a list of ALL of the Paper Titles.
Return to the On-Line Article Review Options Page

Figure 4-4. Artrev.exe.

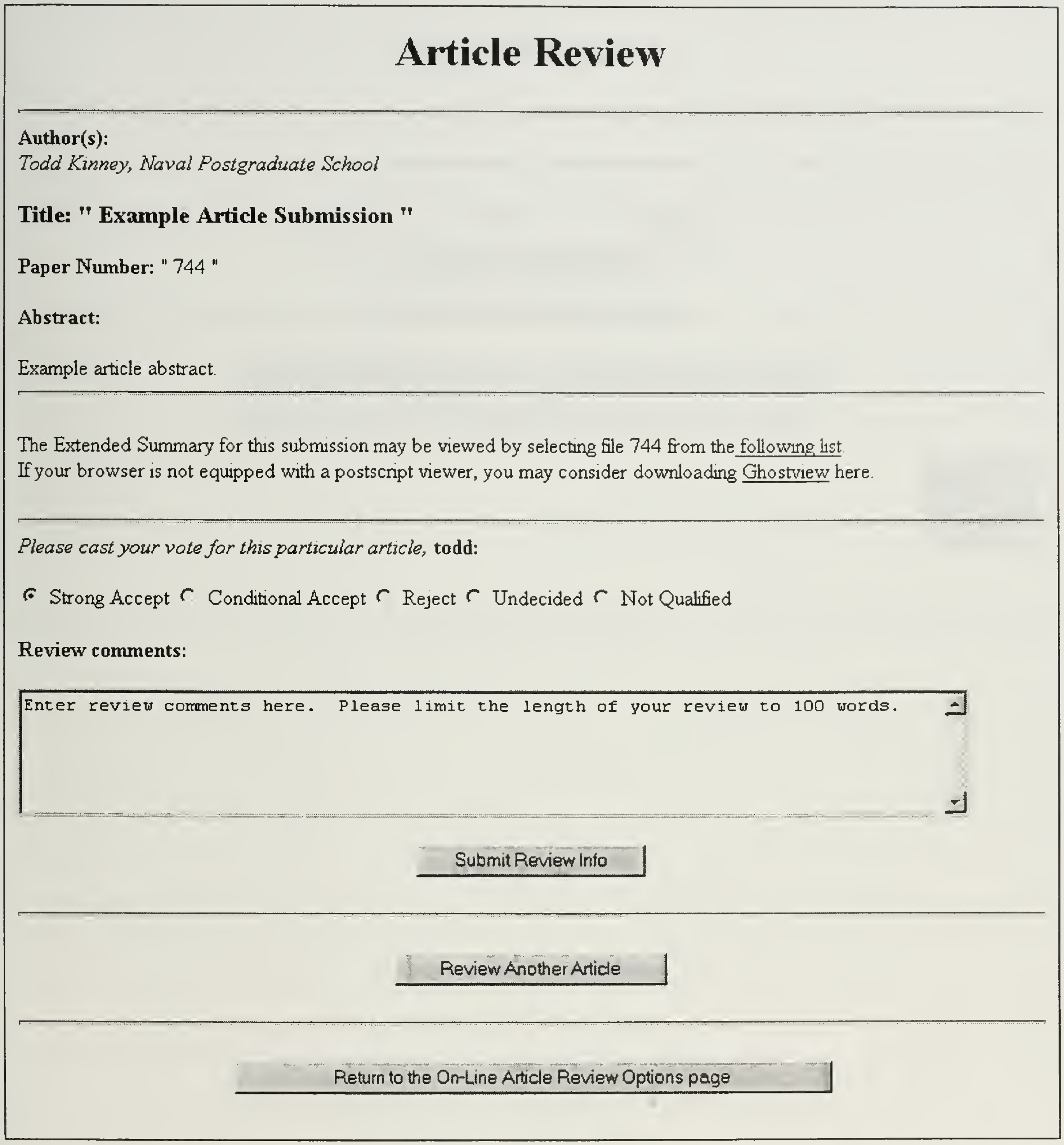

On-Line Article Review System

There are 9 Titles that match your request. Select the article that you would like to review and press submit.

Paper #	Title	Author
◯ 674	A Sonar-based Conducting System for a Virtual MIDI Orchestra	Joohwan Chun
◯ 604	Advanced Filter Design	Brian Evans
◯ 510	Arithmetic Arrays using Cellular Neural Networks	Saeid Sadeghi-Emamchaie
◯ 744	Example Article Submission	Todd Kinney
◯ 616	Lazy Rounding	Paul Fiore
◯ 508	On the Use of Recoding for Low-Power Constant Fixed Point Multiplication	Meng-Jang Lin
◯ 663	Robust High Speed Digital Normal Form Oscillators	Peter Bauer
◯ 487	VHDL SIMULATION, SYNTHESIS AND FPGA IMPLEMENTATION OF FIR FILTERS	K Venkatraman
◯ 452	VHDL-Based Performance Modeling and Its Application to Real Time Infrared Search and Track System Design	Eric Pauer

Select this Paper for Review
Query another Article
Return to the On-Line Article Review Options Page

Figure 4-5. Artsrch2.exe.


Article Review

Author(s):
Todd Kinney, Naval Postgraduate School

Title: " Example Article Submission "

Paper Number: " 744 "

Abstract:

Example article abstract.

The Extended Summary for this submission may be viewed by selecting file 744 from the following list.
If your browser is not equipped with a postscript viewer, you may consider downloading Ghostview here.

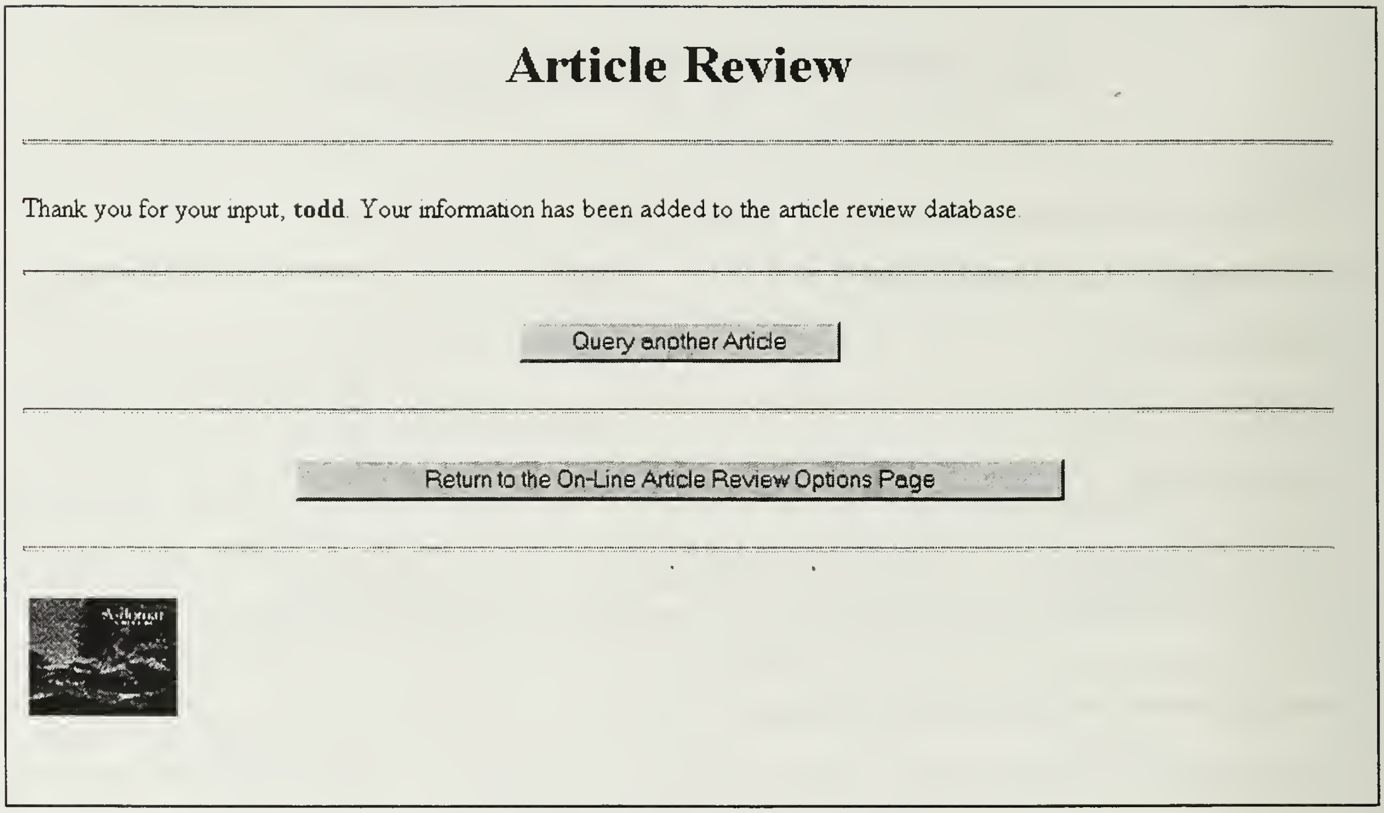
Please cast your vote for this particular article, **todd:**

[✓ Strong Accept] Conditional Accept Reject Undecided Not Qualified

Review comments:

```
Enter review comments here.  Please limit the length of your review to 100 words.
```

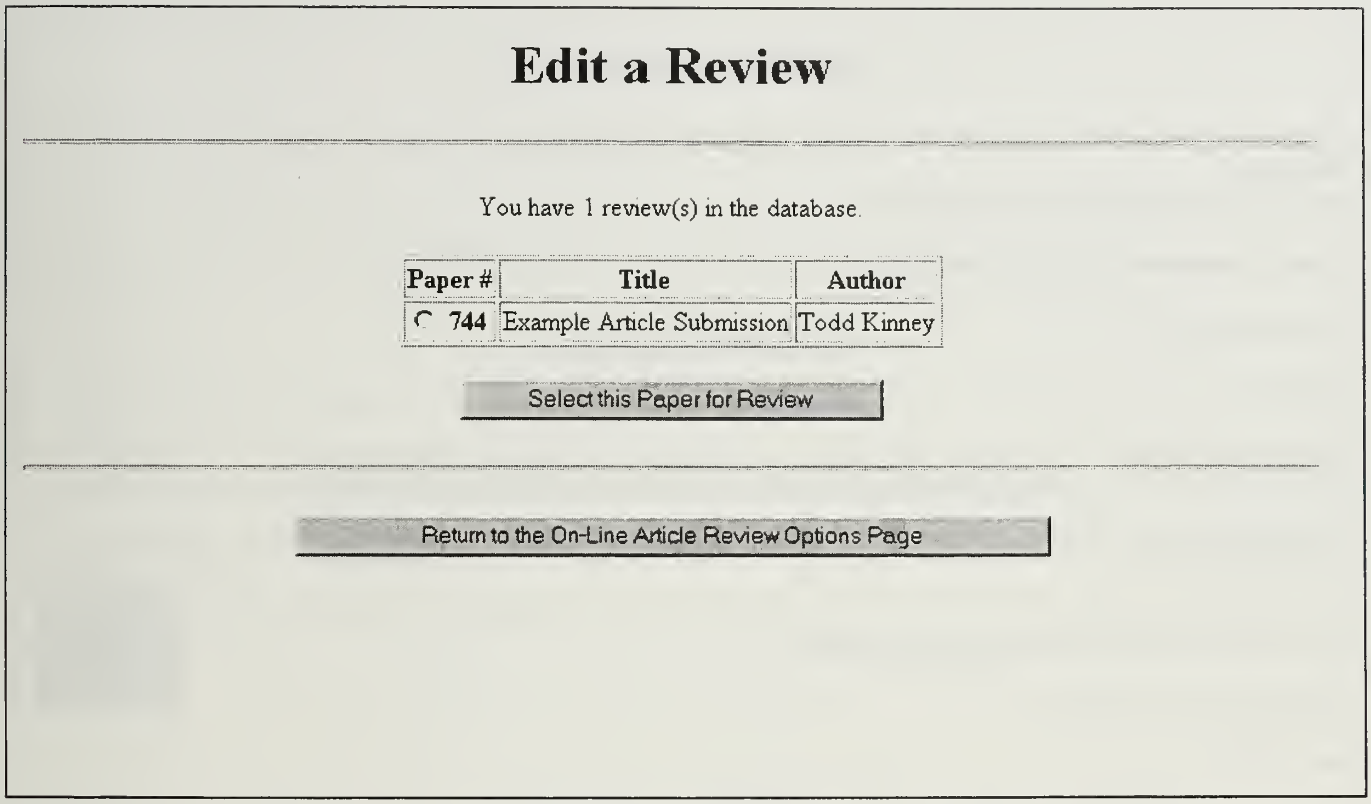
Figure 4-6. Artview2.exe.



Article Review

Thank you for your input, **todd**. Your information has been added to the article review database.

Figure 4-7. Add133.exe.



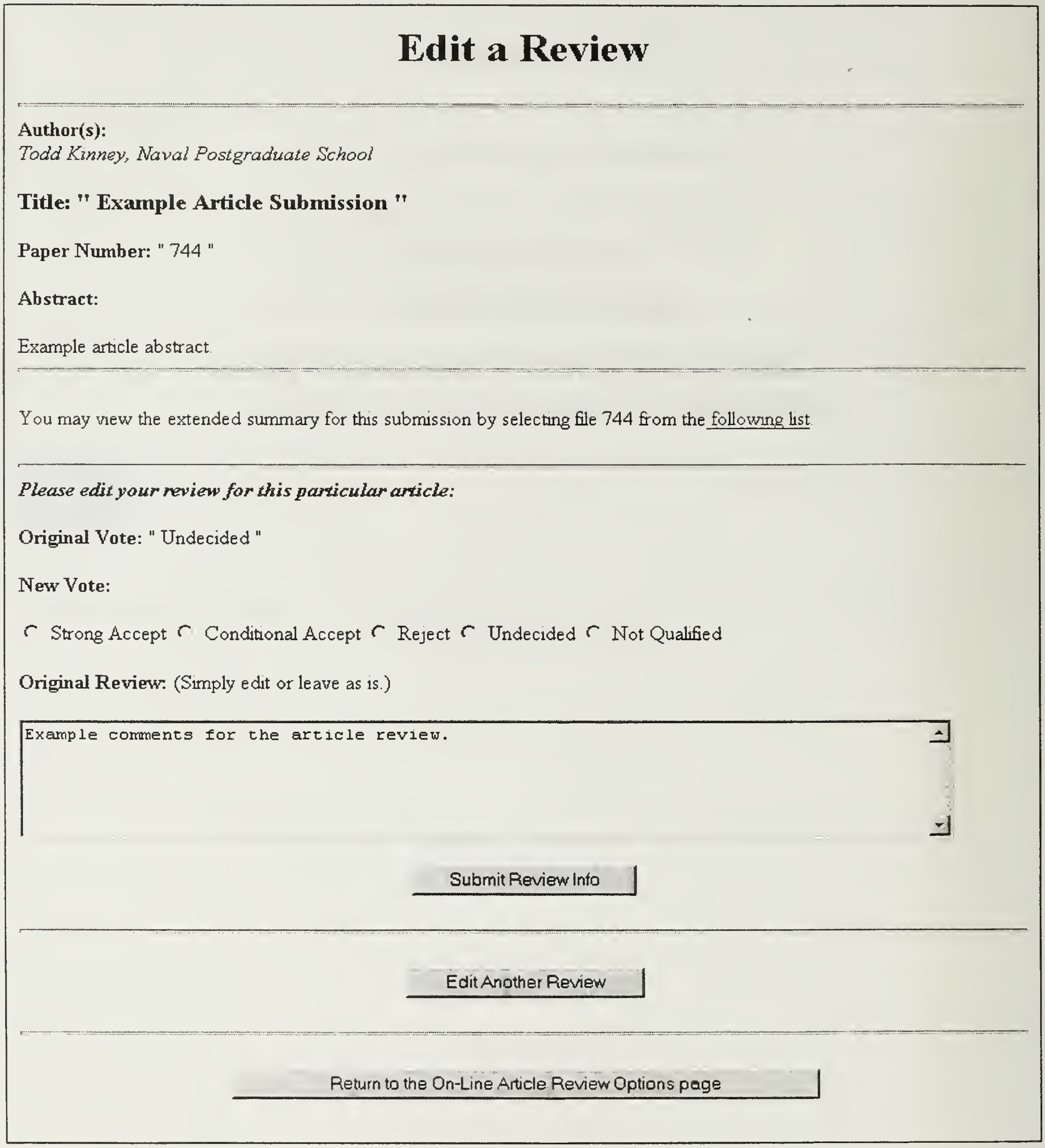
Edit a Review

You have 1 review(s) in the database.

Paper #	Title	Author
◯ 744	Example Article Submission	Todd Kinney

Select this Paper for Review
Return to the On-Line Article Review Options Page

Figure 4-8. Editrev.exe.



Edit a Review

Author(s):
Todd Kinney, Naval Postgraduate School

Title: " Example Article Submission "

Paper Number: " 744 "

Abstract:

Example article abstract.

You may view the extended summary for this submission by selecting file 744 from the following list.

Please edit your review for this particular article:

Original Vote: " Undecided "

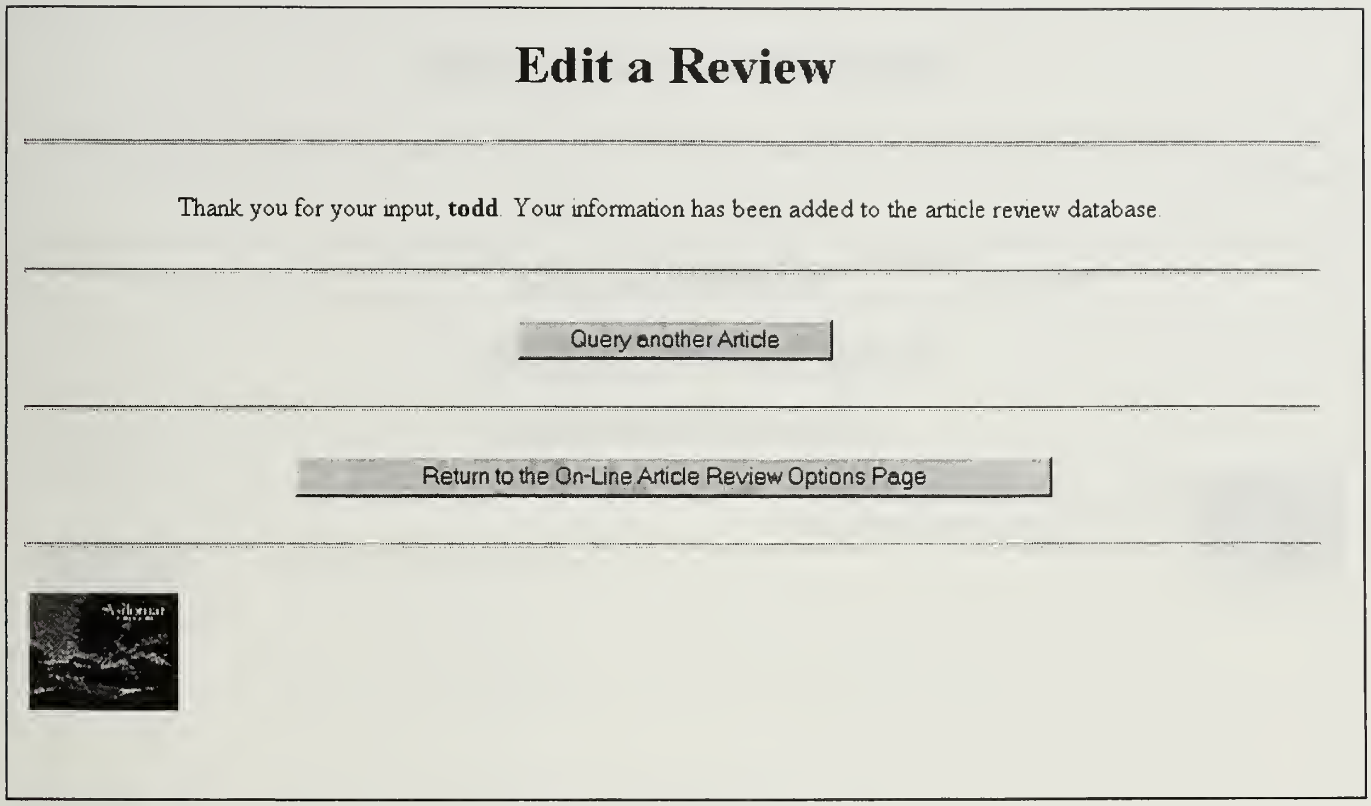
New Vote:

○ Strong Accept ○ Conditional Accept ○ Reject ○ Undecided ○ Not Qualified

Original Review: (Simply edit or leave as is.)

```
Example comments for the article review.
```

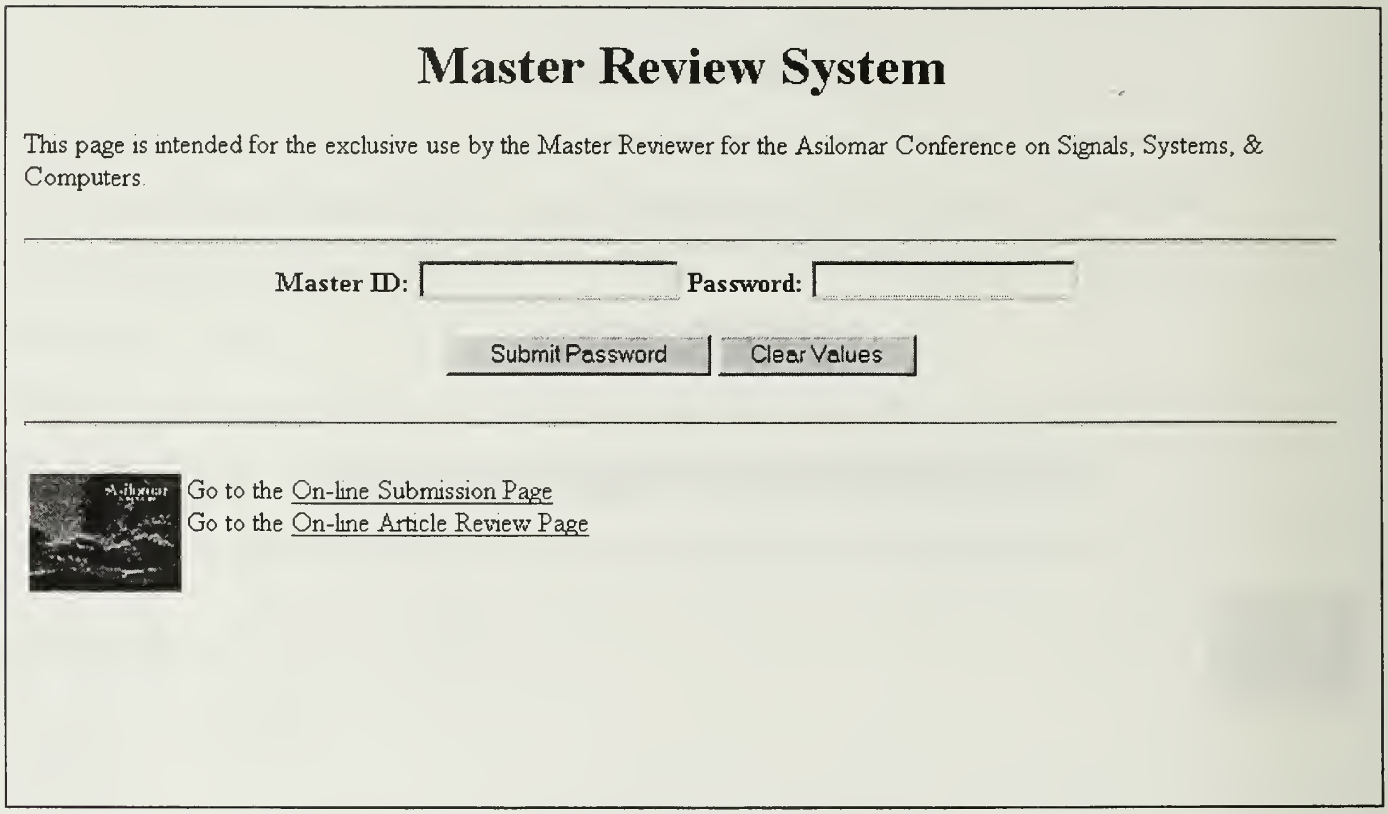
Figure 4-9. Revview.exe.



Edit a Review

Thank you for your input, **todd**. Your information has been added to the article review database.

Figure 4-10. Add133b.exe.



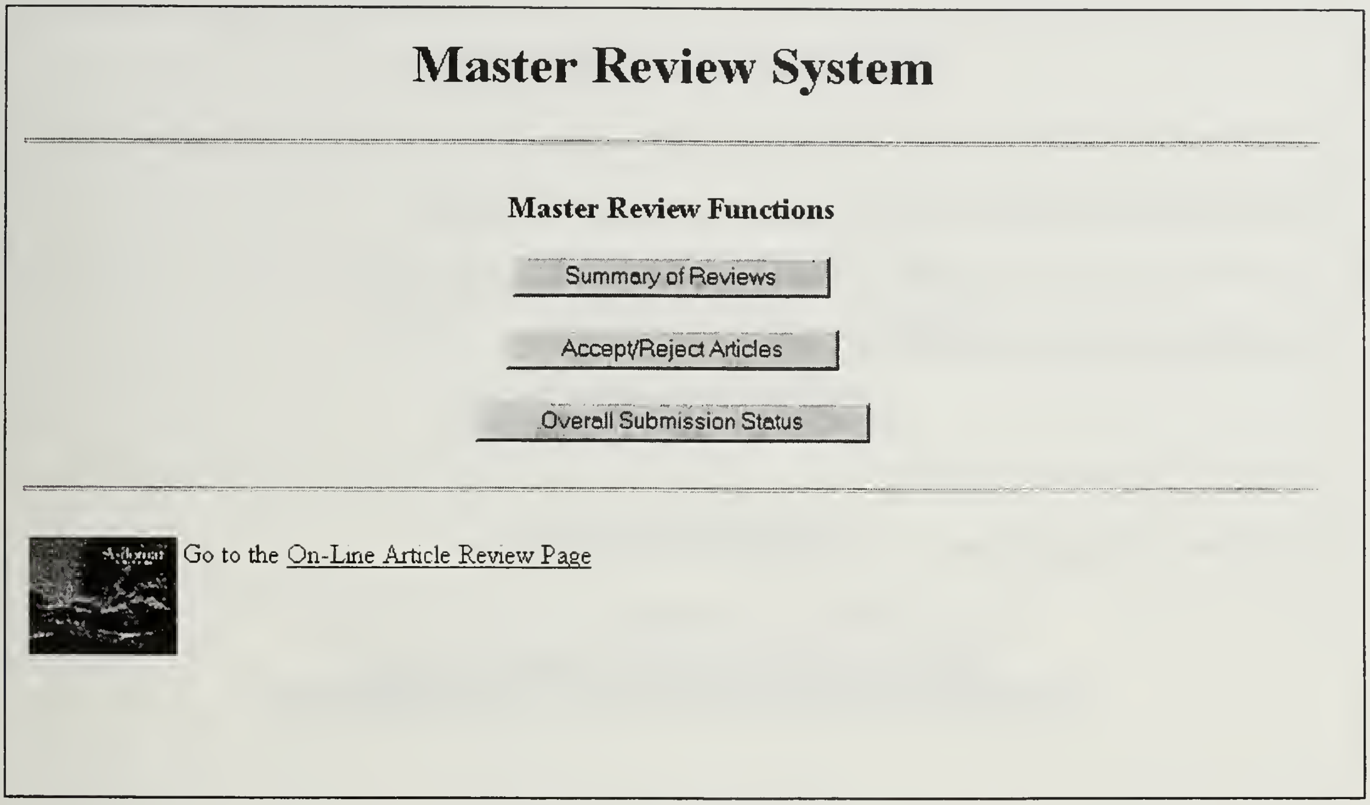
Master Review System

This page is intended for the exclusive use by the Master Reviewer for the Asilomar Conference on Signals, Systems, & Computers.

Master ID: Password:

Go to the On-line Submission Page
Go to the On-line Article Review Page

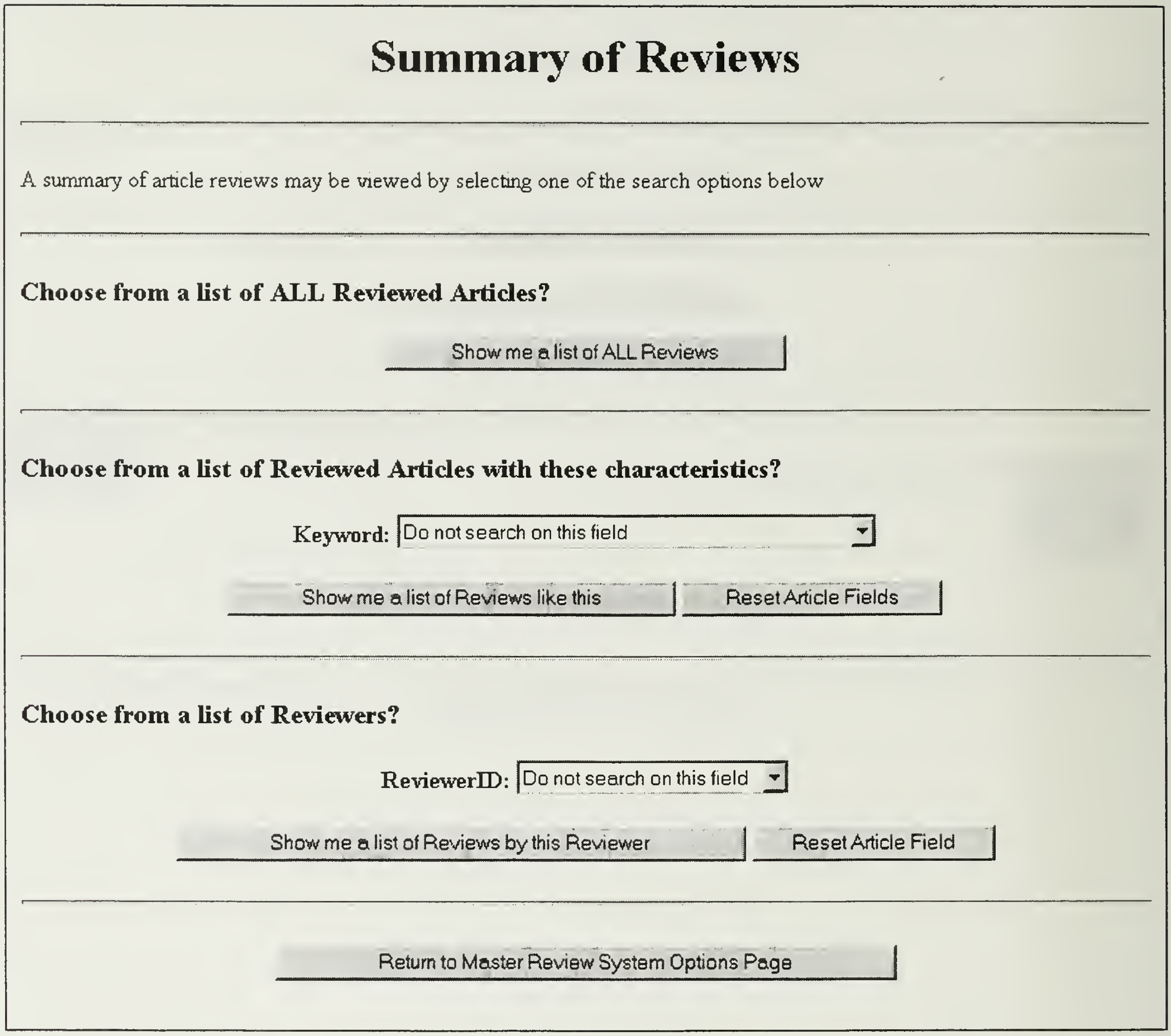
Figure 4-11. Masterev.htm.


Master Review System

Master Review Functions

Go to the On-Line Article Review Page

Figure 4-12. Mastpwd.exe.



Summary of Reviews

A summary of article reviews may be viewed by selecting one of the search options below

Choose from a list of ALL Reviewed Articles?

Show me a list of ALL Reviews

Choose from a list of Reviewed Articles with these characteristics?

Keyword: Do not search on this field

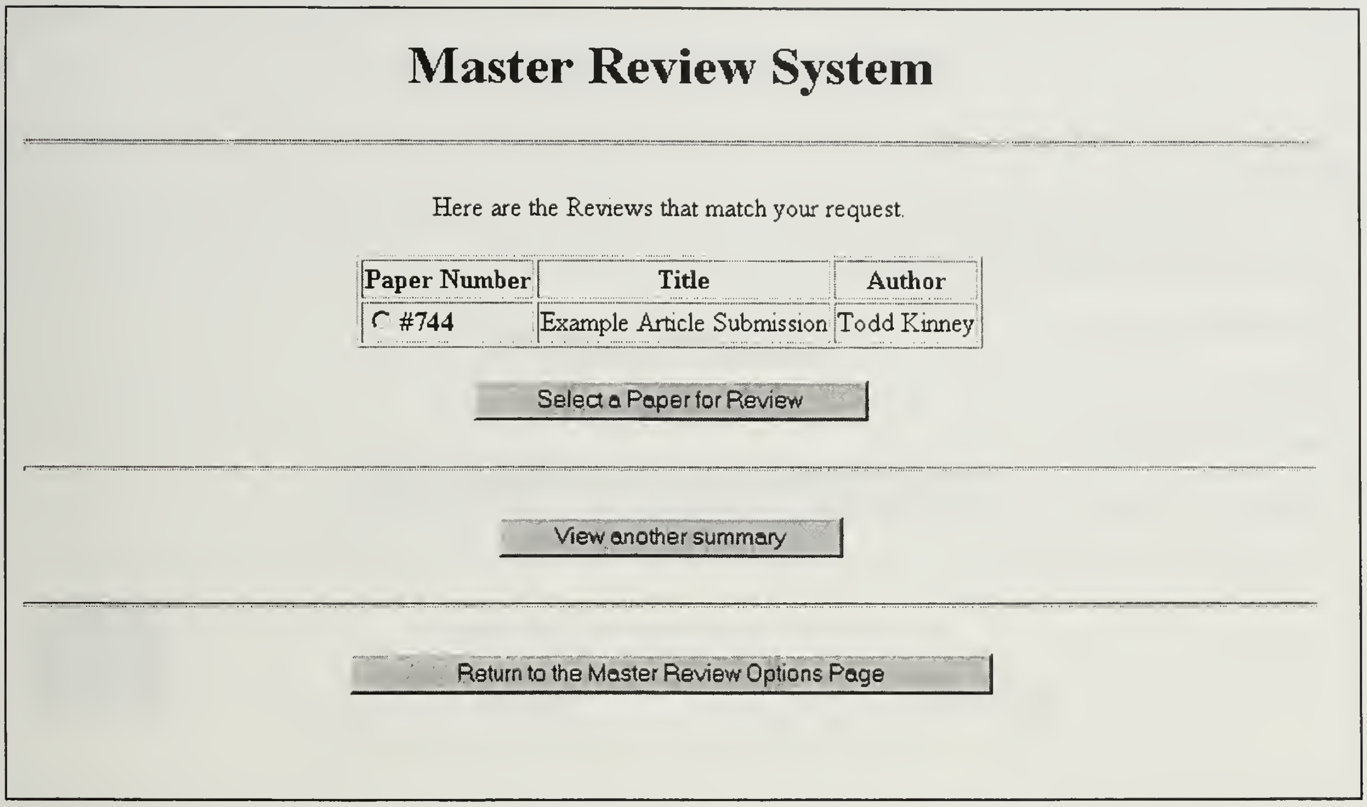
Show me a list of Reviews like this
Reset Article Fields

Choose from a list of Reviewers?

ReviewerID: Do not search on this field

Show me a list of Reviews by this Reviewer
Reset Article Field
Return to Master Review System Options Page

Figure 4-13. Revsum.exe.



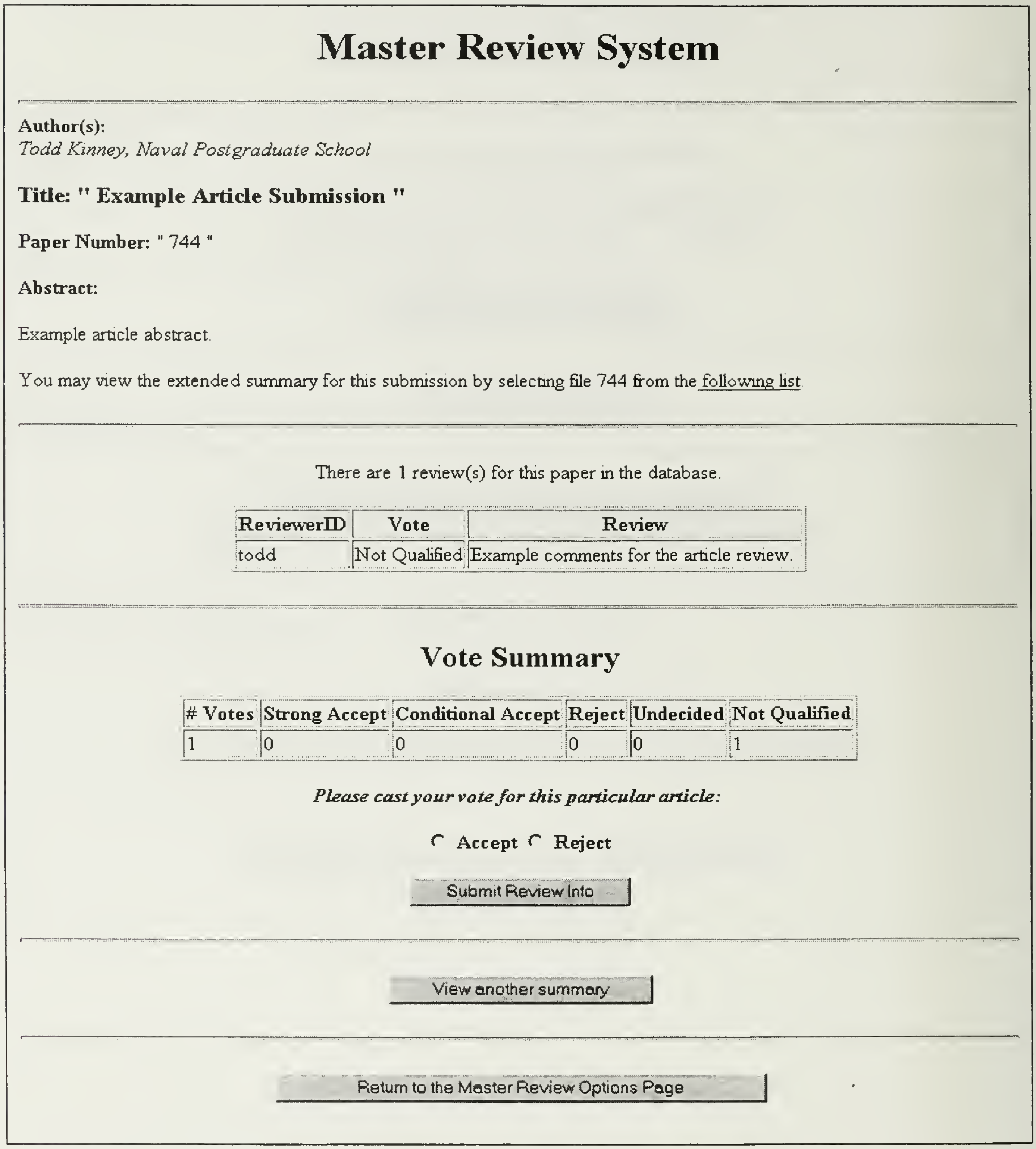
Master Review System

Here are the Reviews that match your request.

Paper Number	Title	Author
◯ #744	Example Article Submission	Todd Kinney

Select a Paper for Review
View another summary
Return to the Master Review Options Page

Figure 4-14. Revsrch.exe.


Master Review System

Author(s):
Todd Kinney, Naval Postgraduate School

Title: " Example Article Submission "

Paper Number: " 744 "

Abstract:

Example article abstract.

You may view the extended summary for this submission by selecting file 744 from the following list

There are 1 review(s) for this paper in the database.

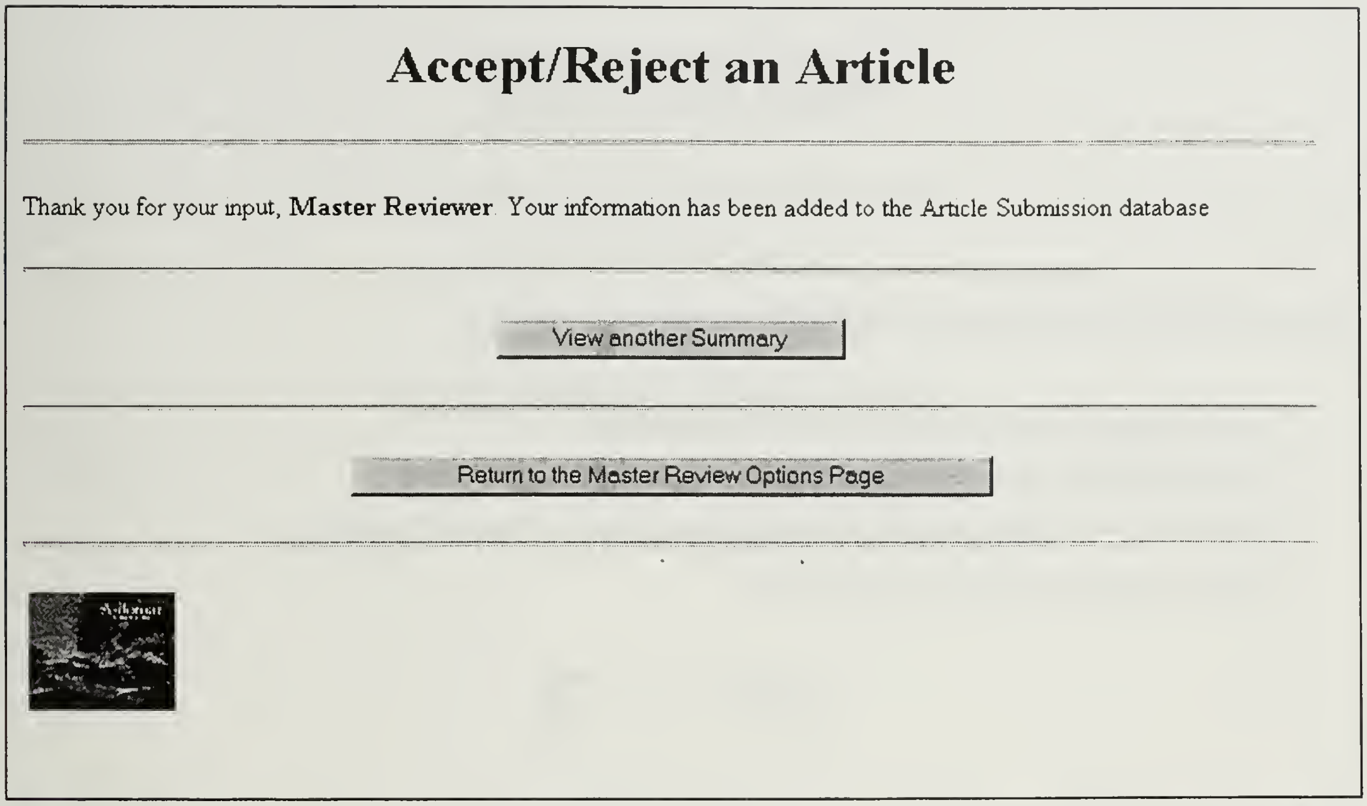
ReviewerID	Vote	Review
todd	Not Qualified	Example comments for the article review.

Vote Summary

# Votes	Strong Accept	Conditional Accept	Reject	Undecided	Not Qualified
1	0	0	0	0	1

Please cast your vote for this particular article:

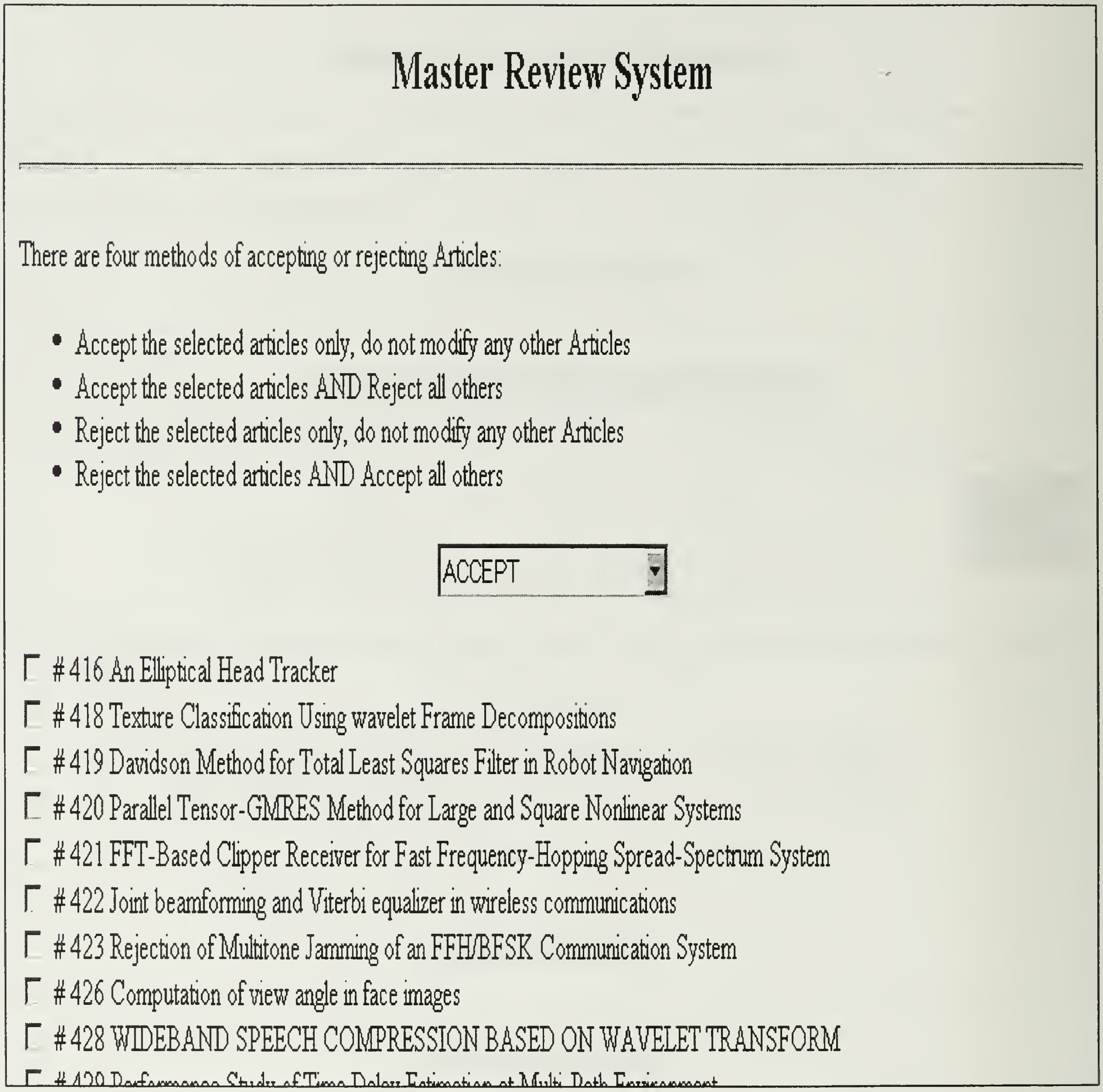
Figure 4-15. Revsrch2.exe.



Accept/Reject an Article

Thank you for your input, **Master Reviewer**. Your information has been added to the Article Submission database

Figure 4-16. MastAcep.exe.


Master Review System

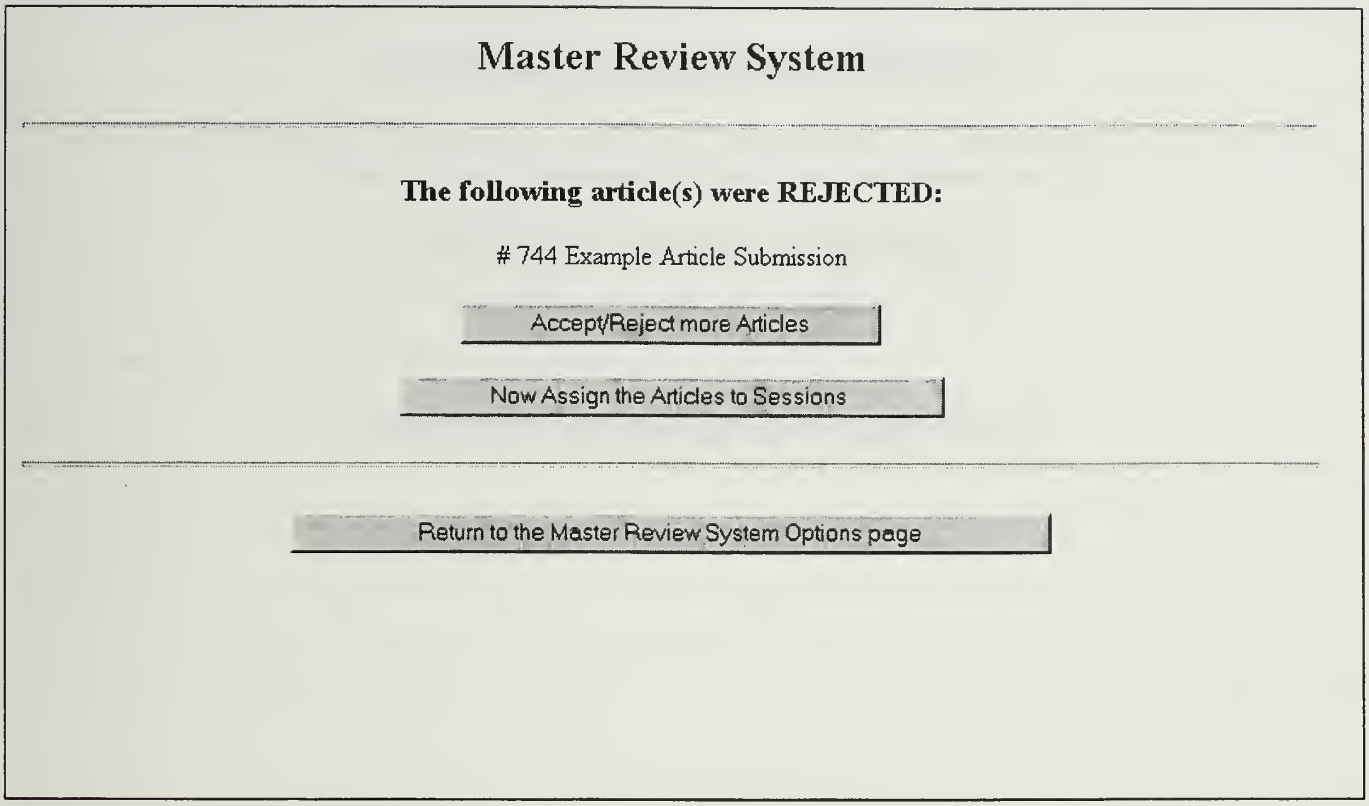
There are four methods of accepting or rejecting Articles:

- Accept the selected articles only, do not modify any other Articles
- Accept the selected articles AND Reject all others
- Reject the selected articles only, do not modify any other Articles
- Reject the selected articles AND Accept all others

ACCEPT

☐ # 416 An Elliptical Head Tracker
☐ # 418 Texture Classification Using wavelet Frame Decompositions
☐ # 419 Davidson Method for Total Least Squares Filter in Robot Navigation
☐ # 420 Parallel Tensor-GMRES Method for Large and Square Nonlinear Systems
☐ # 421 FFT-Based Clipper Receiver for Fast Frequency-Hopping Spread-Spectrum System
☐ # 422 Joint beamforming and Viterbi equalizer in wireless communications
☐ # 423 Rejection of Multitone Jamming of an FFH/BFSK Communication System
☐ # 426 Computation of view angle in face images
☐ # 428 WIDEBAND SPEECH COMPRESSION BASED ON WAVELET TRANSFORM
☐ # 429 Performance Study of Time Delay Estimation at Multi-Path Environment

Figure 4-17. Acceprej.exe.

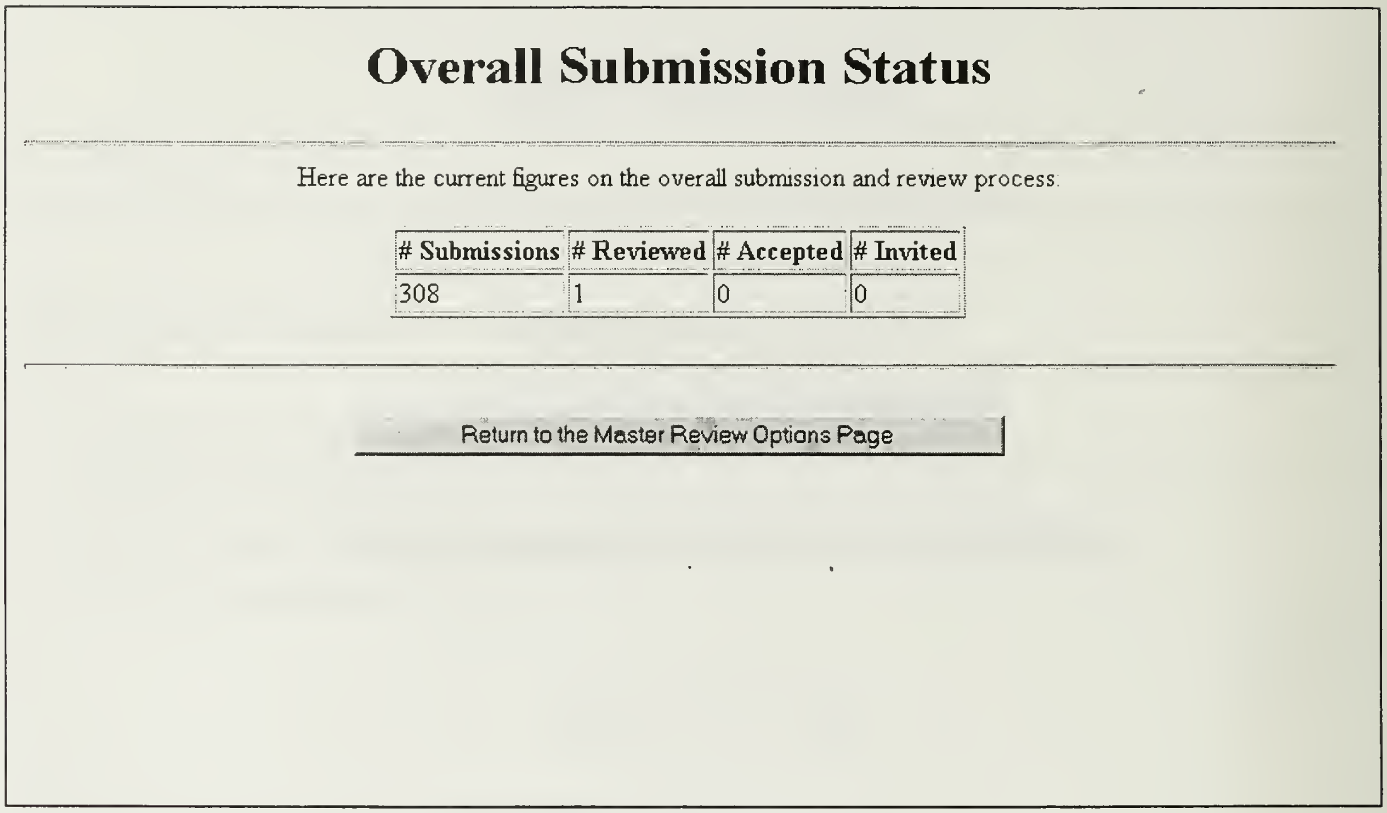

Master Review System

The following article(s) were REJECTED:

# 744 Example Article Submission

Accept/Reject more Articles
Now Assign the Articles to Sessions
Return to the Master Review System Options page

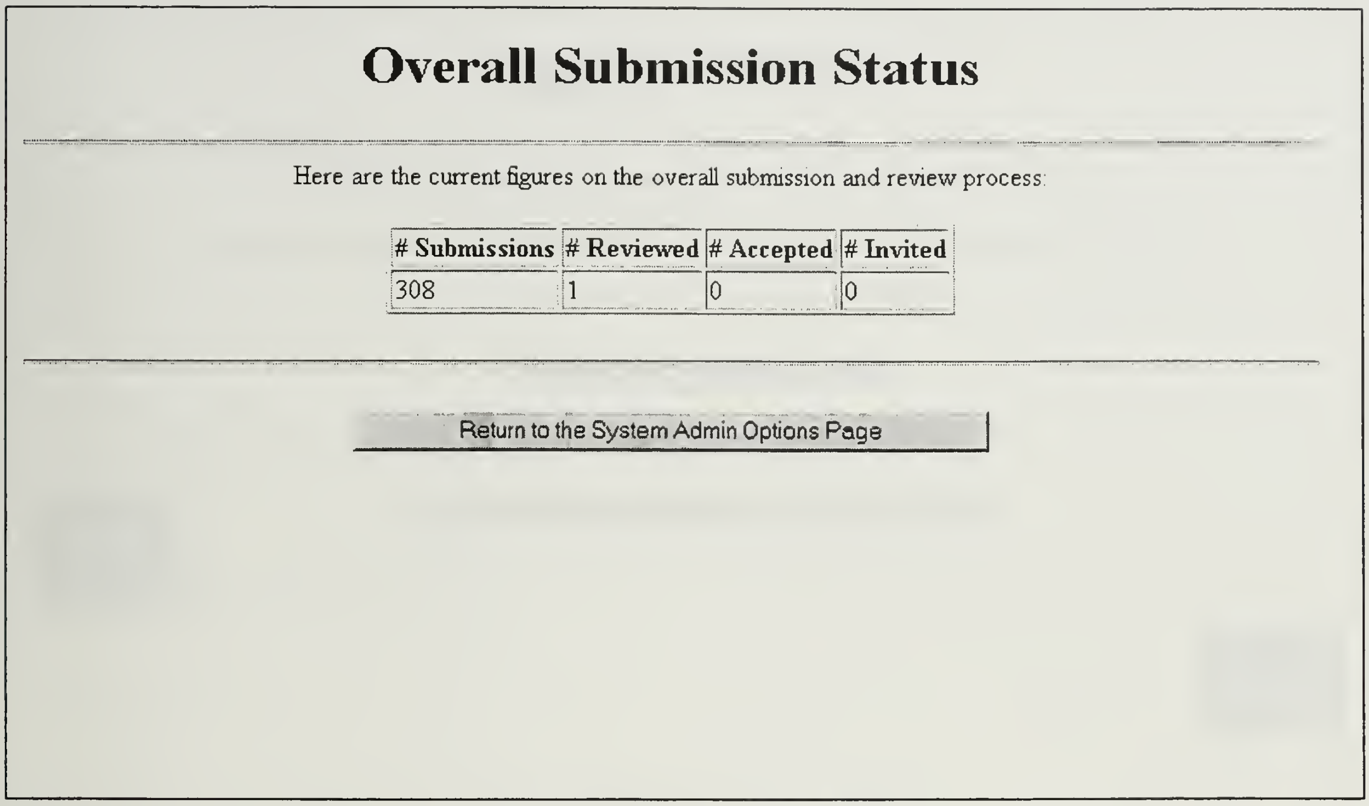
Figure 4-18. Aceprej2.exe.


Overall Submission Status

Here are the current figures on the overall submission and review process:

# Submissions	# Reviewed	# Accepted	# Invited
308	1	0	0

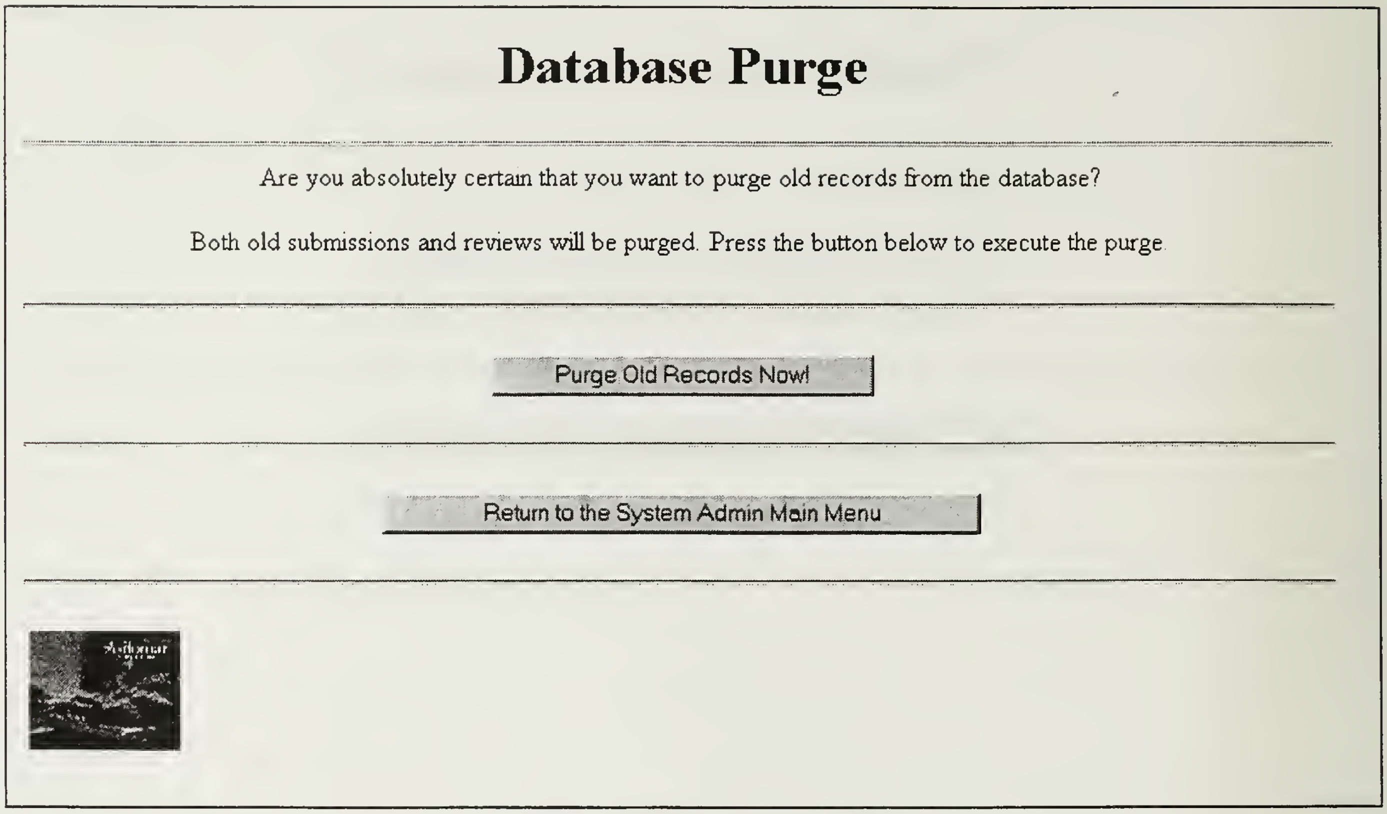
Figure 4-19. Overstat.exe.


Overall Submission Status

Here are the current figures on the overall submission and review process:

# Submissions	# Reviewed	# Accepted	# Invited
308	1	0	0

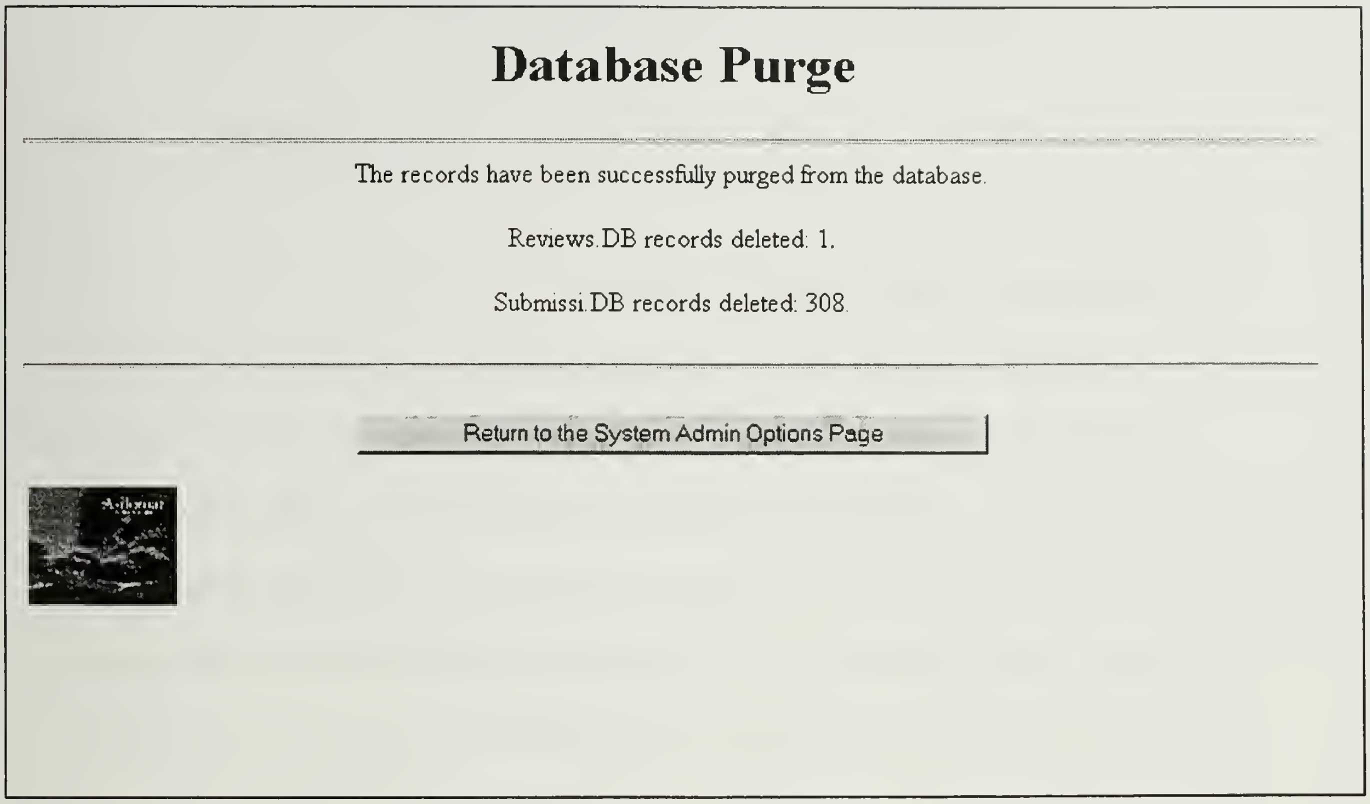
Figure 4-20. Ovrstat2.exe.


Database Purge

Are you absolutely certain that you want to purge old records from the database?

Both old submissions and reviews will be purged. Press the button below to execute the purge.

Figure 4-21. Timeout.exe.



Database Purge

The records have been successfully purged from the database.

Reviews.DB records deleted: 1.

Submissi.DB records deleted: 308.

Figure 4-22. Timeout2.exe.

V. CONCLUSION & RECOMMENDATIONS

A. LOOKING BACK

The Asilomar Conference on Signals, Systems and Computers presents the latest advances in technology. It is only appropriate that the information system of the Conference does the same. The goals of this project were to automate and streamline Conference processes with a Web-based approach to the Conference information system. After careful analysis of the existing system and processes, Coats and Chalfant [Ref. 1] designed and implemented the article submission and system administration portions of the new information system. This thesis saw the design and implementation of the article review portion, as well as the correction and/or modification of the system as a whole as needed. The entirety of the new system was placed "on-line" for use by the 1997 Asilomar Conference in May. Over one hundred authors from around the world utilized the Web-base system to submit their works. Reviewers from across the country have reviewed these same submissions from their own desktops. The initial feedback from those involved has been positive. The system works, is easy to use, and saves time. This project has enabled the Asilomar Conference on Signals, Systems and Computers to move with the advances in information technology into the future. This necessary first step not only improves the way the Conference does business today, it empowers administrators to rethink the way the Conference might be conducted in the future. Global access to Conference information allows Conference functions to be carried out from anywhere in the world with an Internet connection. The time and expense traditionally involved with congregating Conference participants in one geographic location will be a thing of the past.

B. LOOKING AHEAD

1. Recommendations

a. *Scheduling*

The scheduling of selected articles into presentation sessions is a tedious and time consuming task. It is currently conducted by the reviewers themselves. This distracts from their primary task of reviewing articles. By incorporating the automatic scheduling of selected articles into sessions this administrative burden could be removed.

b. *Registration*

The registration of Conference attendees is another function that is currently conducted via regular mail. This results in a significant expense for both participants and administrators of the Conference. On-line registration would not only eliminate mailing expenses, it would also reduce the time and administrative overhead involved with the registration process. This functionality could be developed in-house and added to the current system, depending on the availability of personnel with the necessary technical expertise. A secure on-line payment capability would be required.

WEBSITE Professional 1.0 [Ref. 8] has SSL capability. It would simply be a matter of obtaining a security certificate in order to implement on-line payment via credit cards. Based on the Conference budget, this function could also be outsourced to a commercial company that specializes in on-line registration.

2. Potential DoD Application

The DoD is a large organization with a vast assortment of information needs. Everything from personnel records and financial data to spare parts inventories is maintained in large databases. The diversity in both the geographic location and the stovepipe systems of the various departments and agencies maintaining these databases makes accessing information difficult and/or expensive. These difficulties are precisely why this project was started. The World Wide Web as a communications medium provides viable solutions to difficult information problems. Web based information systems provide organizations with global and seamless access to large databases that is entirely platform independent. Geographic barriers disappear over the Internet, as clients from all over the world can access the system, regardless of what operating system or browser they are using. The barriers of technological diversity also disappear. Standardized Web protocols allow for the universal sharing of data that was once thought to be unattainable. Stovepipe systems once held information locked up in the technological equivalent of a foreign prison. Now that information can be accessed globally and shared universally, thanks to web based information technology. Every mission area, from support to primary combat, that maintains and utilizes data stored in a database can take advantage of the Internet. Time can be saved, expenses can be reduced, and never before thought of functionality can be added to the daily operation of the organization.

C. CLOSING REMARKS

The ultimate goal of this project was to provide conference participants with global access to the various Conference processes. The work done in both this thesis and the initial thesis has accomplished that task. Everything from article submissions and database management to submission reviews can be achieved globally via the Internet.

APPENDIX A. DELPHI TUTORIAL

A. OVERVIEW

Borland's Delphi is the scripting language that was used to create this project's CGI programs. This section provides a brief overview of how these programs were developed. The following section provides an example of a simple CGI program and its code.

Although Delphi is primarily known as a "drag-and-drop" Windows programming tool, it is easily adapted to the construction of interactive WWW scripts. Components are the building blocks that make this possible in Delphi. A component is an object or module that provides some specific bit of functionality to an object oriented program. A CGI component provides the CGI functionality that turns forms into interactive executables. At the start of this project, the field of dynamic Web page development was in its infancy. The number of commercially available CGI components for Delphi was limited. As a result, much of the CGI functionality had to be manually coded into the Delphi programs. This resulted in a time consuming effort that could stand some automation in order to take full advantage of Delphi's object oriented drag-and-drop characteristics. Fortunately, one knows that wherever there is a market, products will appear. As a result, commercial vendors have since produced a number of CGI components that eliminate the need for manual coding in Delphi. They provide CGI functionality by simply "dragging and dropping" a component object onto a form, vice manually typing code into the text editor.

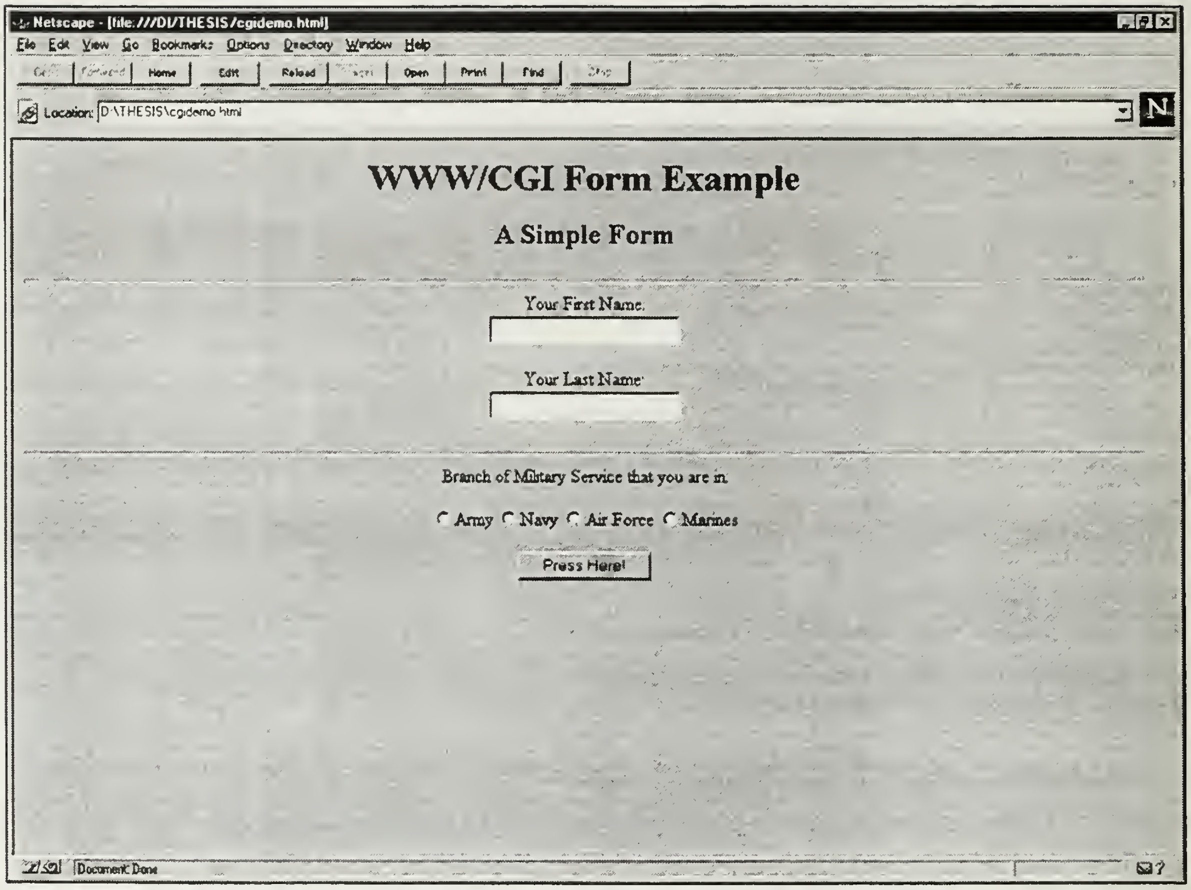
As mentioned above, Delphi components or modules contain Object Pascal coding for the particular functionality that they provide. In the case of CGI components, the objects contain CGI scripts that allow the Web server to communicate with a browser via an HTML document or form. Both the HTML code for the requesting form and the CGI script for the task the form is requesting are located and compiled within the Delphi program. Web servers maintain all CGI executables or, in this case, Delphi programs in specially designated directories. For example, a server might have the following sub-directories for its CGI programs: 1) cgi-bin, for Windows Perl CGI scripts, 2) cgi-win, for other Windows CGI scripts, and 3) cgi-dos, for DOS Perl CGI scripts.

Once the project Web pages and database queries are planned, Delphi programs are designed and compiled as necessary. The executables may be set-up to be automatically stored in WEBSITE's cgi-win directory upon compiling. Any program stored in this directory is ready for immediate use by the server when specifically called upon by a user.

B. SAMPLE APPLICATION

The following CGI example is a simple form that solicits user input information that will be gathered by the CGI script and stored in a database on the Web server. Figure A-1 shows the HTML form that collects this input data. The HTML code for the form appears below.   Explanatory statements appear in the format < -- comments -- >.

```
<HTML>
<HEAD>
<TITLE>WWW/CGI Form Example</TITLE>
<CENTER><H1>WWW/CGI Form Example</H1></CENTER>
<HR>
</HEAD>
<BODY>
<FORM ACTION="/cgi-win/cgidemo.exe". METHOD="POST">
Your First Name:<BR>
<INPUT TYPE="text" NAME="firstname"><P>
Your Last Name:<BR>
<INPUT TYPE="text" NAME="lastname"><P>
<HR>
Branch of Military Service that you are in:<P>
<INPUT TYPE="radio"  NAME="branchservice"  VALUE="Army">Army
<INPUT TYPE="radio"  NAME="branchservice"  VALUE="Navy">Navy
<INPUT TYPE="radio"  NAME="branchservice"  VALUE="Air Force">Air Force
<INPUT TYPE="radio"  NAME="branchservice"  VALUE="Marines">Marines<P>
<INPUT TYPE="submit"  VALUE="Press Here!">
</FORM>
</BODY>
</HTML>
```


Netscape - [file:///D|/THESIS/cgidemo.html]

Location: D:\THESIS\cgidemo.html

WWW/CGI Form Example

A Simple Form

Your First Name:

Your Last Name:

Branch of Military Service that you are in:

Army Navy Air Force Marines

Press Here!

Document: Done

File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop

Figure A- 1

The next step is to create the CGI program using Delphi's Integrated Development Environment (IDE).

1. To load the IDE click on the icon that resembles a house with columns
2. You should be presented with a new project. If not select "New" from the "File" menu
3. On the Component Palette, click on the CGI speed tab, click on the first CGI icon, then click in the form window. The first CGI icon (CGIEnvData) loads general scripting information to your form, the second (CGIDB) adds database specific components which will not be used in this example.

4. The CGI element icon should have black boxes in the corners to indicate that it is selected. Look at the Object Inspector window. The 'properties' tab should be selected. All of the default values are acceptable. If you click on the 'events' tab you should not see any available events.
5. Click on a blank spot in the form window to deselect the CGI element. Now let's look at the Object Inspector window. Make sure that the 'properties' tab is selected. For this example, all of the default values are acceptable. These values and their meanings are covered in section three of this tutorial.
6. Now, click on the 'events' tab in the object inspector. This is a list of things that will happen when the user initiates certain actions. I want this application to create a form when it is called, so click on the box next to "OnCreate". Now type a name for a procedure which will create a form (I used the name FormCreate in the example. You may use any number of characters, but no spaces). After you type the name, hit 'enter', this will bring up the code editor window. The following code will already have been added for you:

```
----------------------------------------------------

unit Main;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, IniFiles, CGI, DB;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
```

```
end;
end.
---------------------------------------------------
```

7. The cursor will be after the begin statement. You will now write the code that will a) get data from the HTML form, and b) create the new page using that data.
8. The first step is to declare variables that the Delphi application will use. The variable declaration section comes before the 'begin' statement and after the 'procedure' statement. You will need to declare one variable for each 'INPUT NAME' or 'SELECT NAME' in the HTML document. Note: to avoid confusion, use different variable names than what 'NAMES' were in the HTML document. Since we are reading in text from the form, the variables will be set to type 'string'. The variable declarations section for this example is as follows:

```
---------------------------------------------------
var
  FirstName : String;
  LastName : String;
  Branchservice : String;
---------------------------------------------------
```

9. Now move the cursor past the 'begin' statement. Since this procedure uses CGIEnvData, we must add the line: with CGIEnvData1 do The '1' on the end tells Delphi that this will be the first instance of CGIEnvData.
10. Now on a new line type: *begin* This starts the procedure to gather and use the data.
11. After this second 'begin' statement, three statements must be added which are required in ALL WEBSITE Delphi applications. They are:

```
webSiteINIFilename := paramstr(1);
application.onException := cgiErrorHandler;
application.processMessages;
```

12. The next line to be added: createStdout; tells Website that an HTML form will be created.
13. The next line is: sendPrologue; This statement is accomplished before the creation of the new HTML document and is the beginning of the section where you will gather the user's data. In the following section, you will be setting the Delphi variables you declared earlier, equal to the 'INPUT NAMES' from the form. The format for that is:

    variable := getSmallField('INPUT NAME' or 'SELECT NAME');

    In the example this section was as follows:

```
FirstName := getSmallField('firstname');
LastName := getSmallField('lastname');
Branchservice := getSmallField('branchservice');
```

14. Now that you have collected the data, you will create the new HTML document via the ‘send’ command. The general format is:
    send(‘<HTML tag>’ + variables + ‘regular text’);
    Note that items within the ( ) can be in any order, so long as proper HTML is used. The key here is that variables may be included and are indicated by not being enclosed in quotes. There are a few special ‘send’ commands such as ‘sendHR’ and ‘sendTitle’ that should be self-explanatory.
15. The following is a completion of the new HTML document:

```
-----------------------------------------------

send('<HTML><HEAD>');
send( '<TITLE>WWW/CGI Example</TITLE></HEAD>');
send('<BODY BGCOLOR="#FFFFFF"><HR>Hello ' + FirstName + ' ' + LastName + ',
<P>');
send('The Request Method is <B>' + RequestMethod + '</B>.<BR>');
send('The Server Name is <B>' + ServerName + '</B>.<BR>');
send('Your IP Address is <B>' + RemoteAddress + '</B>.<P>');
send('Sir, the time on deck is <B>' + DateTimeToStr(Now)+ '</B><P><HR>');
send('You are employed by the <B>' + Branchservice + '</B>.');
send('</BODY></HTML>');

-----------------------------------------------
```

16. Now that the new form is complete you must tell Website. The next line is:
    closeStdout;
17. Then: end;
18. Now the applications opened just after the ‘begin’ statement must be closed via the command: closeApp(application);
19. Now another: end; to close the procedure
20. And one more: end; to close the entire application
21. Now let’s save the project. Select File | Save project as
22. Delphi will prompt you to enter a name for the code you have just written. This name must be different than the name of the executable file which Delphi asks for next. For instance, if you planned to have the executable named “Madlib.exe” as in this example, you should name the .PAS (short for Pascal no doubt) something else. I used “Madlibpr.pas”. Now Delphi will ask for your project name. This is the time to enter the executable program’s name.
23. You are now ready to compile! From the top menus, select Options | Project. The ‘Project Options’ window will open. On the quick tabs, select ‘Directories/Conditionals’. The field ‘Output Directory’ is blank. This field indicates the directory where you want the compiled executable to be placed. Click on the down arrow to the right of the field and you will be given a choice of directories that you have previously entered. ‘C:\website\cgi-win’ is the directory for Windows CGI applications, so select that from the list or type it in manually. Click ‘OK’.

24. Now from the menu, select Compile | Compile. If you have any errors in your code, they will be highlighted in red. A vaguely helpful message will be displayed at the bottom of the window. Note that semi-colons or the absence of semi-colons is very important. One misplaced semi-colon can cause many errors. Generally spaces are not important to Delphi. Also notice that some words become bold when you type them (i.e. **var, procedure, begin,** etc.) these words are key words and may not be used elsewhere in your script.
25. Once your code has compiled, you are done. Exit Delphi and fire up your web browser. A complete copy of the example code follows on the next two pages.

```
unit Main;
interface
uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, IniFiles, CGI, DB;
type
 TForm1 = class(TForm)
 CGIEnvData1:TCGIEnvData;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
var
  FirstName : String;
  LastName : String;
  Branchservice : String;
begin
  with CGIEnvData1 do begin
    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

   createStdout;
   sendPrologue;

  {Retrieve the user input from the previous form.}

  FirstName := getSmallField('firstname');
  LastName := getSmallField('lastname');
  Branchservice := getSmallField('branchservice');


  send('<HTML><HEAD>');
  send( '<TITLE>WWW/CGI Example</TITLE></HEAD>');
  send('<BODY BGCOLOR="#FFFFFF"><HR>Hello ' + FirstName + ' ' + LastName + ',
<P>');
   send('The Request Method is <B>' + RequestMethod + '</B>.<BR>');
   send('The Server Name is <B>' + ServerName + '</B>.<BR>');
  send('Your IP Address is <B>' + RemoteAddress + '</B>.<P>');
  send('Sir, the time on deck is <B>' + DateTimeToStr(Now)+ '</B><P><HR>');
  send('You are employed by the <B>' + Branchservice + '</B>.');
  send('</BODY></HTML>');

  closeStdout;
  end;
closeApp(application);
end;
end.
```

This example demonstrates, to a modest degree, the amount of manual coding required in developing CGI programs using Delphi. Although this may appear overwhelming at first, the actual operation of Delphi becomes fairly easy with practice. Most of the actions discussed in this example are common to Delphi CGI programing. The layout is most always the same, with variable declarations coming first, followed by standard WEBSITE statements, followed by prologue operations, followed by Web page creation.

The object oriented programming language used by Delphi is Object Pascal, a close relative of Pascal. General programming techniques were not included in this discussion. It is assumed that the Delphi user has some programming knowledge. Also, Delphi comes with an extensive online help system that covers many programming basics. Delphi also has built-in interactive tutors which include a demonstration of how to build an application (non-CGI only).

APPENDIX B. DATABASE TABLES

There were three database tables designed for this portion of the project. Salsa, a shareware database design tool [Ref. 13], was used in their design. The first two tables, **Review.DB** and **Master.DB**, are used in the identification and authentication of reviewers and the Master Reviewer, respectively. The **Reviews.DB** table is used for storing article review data generated during the review process. The Semantic Object Diagram for these three new tables is depicted in Figure B-1.

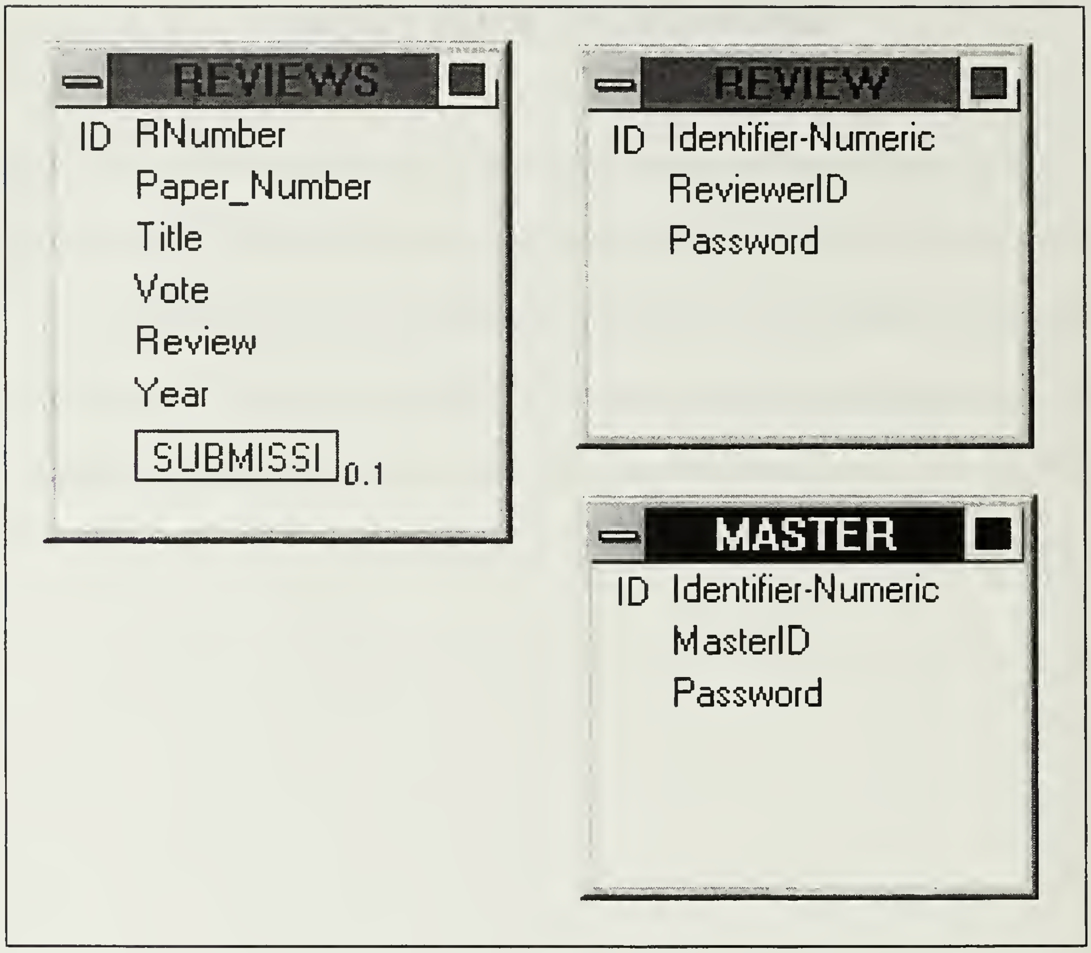

REVIEWS
ID RNumber
Paper_Number
Title
Vote
Review
Year
SUBMISSI 0.1
REVIEW
ID Identifier-Numeric
ReviewerID
Password
MASTER
ID Identifier-Numeric
MasterID
Password

Figure B-1

APPENDIX C. ON-LINE ARTICLE REVIEW SUB-SYSTEM CODE

This appendix contains the HTML and Delphi code for the HTML documents and Delphi executables that comprise the On-line Article Review Sub-system. The relationships of the programs are depicted in Figure C-1.

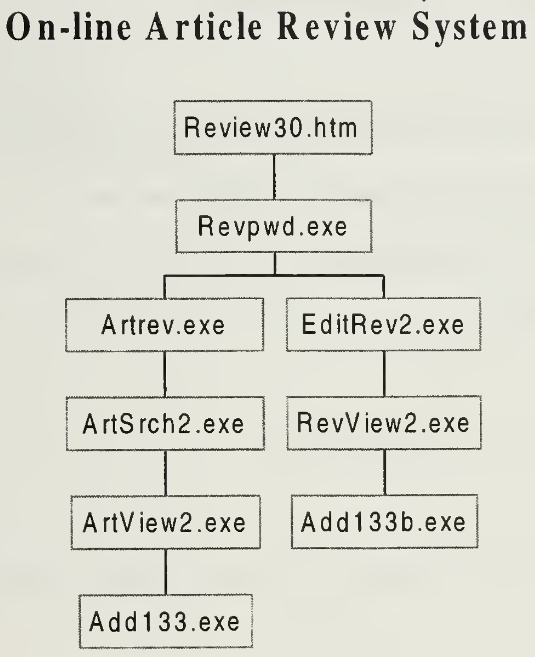


Figure C - 1.

The On-line Article Review System log-on page: **Review30.htm**

```
<HTML>

<HEAD>

<TITLE>On-line Article Review System</TITLE>

</HEAD>

<BODY bgcolor=FFFFFF>

<center><H1>On-line Article Review System</center></H1>

<P> This page is intended for the exclusive use of Reviewers for

the Asilomar Conference on Signals, Systems, & Computers.

<FORM ACTION="../cgi-win/Revpwd.exe " METHOD="POST">

<HR><CENTER>

<B>Reviewer ID:    </B><INPUT NAME="ReviewerID" Size="25" TYPE="text">

<B>Password:    </B><INPUT NAME="pwd" Size="20" TYPE="password">

<P><INPUT TYPE="submit" Value="Submit Password">  <INPUT TYPE="reset"'

VALUE="Clear Values"></CENTER>

<P><hr><P>

<IMG SRC="asil2b0.gif " align=left alt="Asilomar facility">

Go to the <A

HREF="http://XXX.XXX.XXX.XXX/submit/Index.htm">On-line Submission Page</A><BR>

</BODY>

</HTML>
```

```
unit Rev_pwd;

{This is the main menu form for the Review System.  It allows reviewers to review articles and to edit
reviews.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    Query1: TQuery;
    DataSource1: TDataSource;
    Table1: TTable;
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    Query1Password: TStringField;
    Table1ReviewerID: TStringField;
    Table1Password: TStringField;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  dReviewerID: string;
  dpwd: string;
  dbpassword: string;
  dflag: string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

    { Get fields from search page }
  dReviewerID:= getsmallfield ('ReviewerID');
  dpwd:= getsmallfield ('pwd');
  dflag:= getsmallfield ('flag');

  send ('<HTML><HEAD>');
  SendTitle('On-line Article Review System');     { Every page gets this }
  send ('</HEAD><BODY bgcolor=FFFFFF>');

  end;

{Retrieve Password of the given User}
with Query1 do begin

  close;
  sql.clear;
  sql.add('Select * FROM Review WHERE ReviewerID = "' + dReviewerID + '"');
  open;

  Table1.Open;
  Table1.First;
  dbpassword := fieldByName('Password').Asstring;
  Table1.Close;
  end;

with CGIEnvData1 do begin

   {Determine whether password was correct}
   if (dbpassword = dpwd) or (dflag = '1') then begin    {password correct or flag set}

   send('<center><H1>On-line Article Review System</H1></center>');
   sendhr;
   send('<center><h3>Review Functions</h3></center>');

   {Review abstracts data Button}
   send('<P><FORM ACTION="../cgi-win/Artrev.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
   send('<P><center><INPUT TYPE="submit" Value="Review an Abstract">
   </center></FORM>');

   {Edit a Review button}
   send('<P><FORM ACTION="../cgi-win/Editrev.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
   send('<P><center><INPUT TYPE="submit" Value="Edit a Review">
   </center></FORM>');
   end

   else begin     {Password **incorrect**}

   send('<center><H1>On-Line Article Review System</H1></center>');
   sendhr;
   send('<center><H2>Your password was <strong>not accepted!!</strong>
   </H2></center>');
   send ('<P>Please ensure that you are authorized to access this information.');
```

```
    send ('<P>If you made an error, then please <A HREF="../review/review30.htm">try
    to login again.</A>');

    send ('<P>If you are not authorized to access this information, please note that a log is');
    send ('maintained that includes your URL.  You may be subject to civil and/or
    criminal');
    send ('prosecution for your actions.<BR>');

    end;  {flag incorrect}

    send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar
    facility">');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );

      end; {with cgiEnvData1 do}

   end;   {FormCreate}

end.
```

The Article Review Search Menu: **Artrev.exe**

```
unit Art_rev;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Query1: TQuery;
    Table1: TTable;
    Table2: TTable;
    Table2ANumber: TIntegerField;
    Table2FName: TStringField;
    Table2LName: TStringField;
    Table2Initial: TStringField;
    Table2Honorific: TStringField;
    Table2Institution: TStringField;
    Table2Department: TStringField;
    Table2Mailstop: TStringField;
    Table2Address_Street: TStringField;
    Table2Address_City: TStringField;
    Table2Address_State: TStringField;
    Table2Address_Zip: TStringField;
    Table2Country: TStringField;
    Table2Phone_LocalNumber: TStringField;
    Table2Phone_AreaCode: TStringField;
    Table2Phone_FaxNumber: TStringField;
    Table2Email: TStringField;
    Table3: TTable;
    Query1Number: TIntegerField;
    Table3Number: TIntegerField;
    Table3Name: TStringField;
    Table1Paper_Number: TIntegerField;
    Table1Title: TStringField;
    Table1Invited: TStringField;
    Table1Accepted: TStringField;
    Table1ContactAuthorNumber: TIntegerField;
    Table1Session: TStringField;
    Table1OrderInSession: TStringField;
    Table1PresentationTime: TStringField;
    Table1Keyword1: TStringField;
    Table1Keyword2: TStringField;
    Table1Keyword3: TStringField;
    Table1Abstract: TMemoField;
    Table1ContactOrder: TStringField;
    Table1FName2: TStringField;
    Table1LName2: TStringField;
    Table1Initial2: TStringField;
    Table1Institution2: TStringField;
    Table1Order2: TStringField;
    Table1FName3: TStringField;
    Table1LName3: TStringField;
    Table1Initial3: TStringField;
    Table1Institution3: TStringField;
    Table1Order3: TStringField;
    Table1FName4: TStringField;
    Table1LName4: TStringField;
    Table1Initial4: TStringField;
    Table1Institution4: TStringField;
    Table1Order4: TStringField;
    Table1FName5: TStringField;
    Table1LName5: TStringField;
    Table1Initial5: TStringField;
    Table1Institution5: TStringField;
    Table1Order5: TStringField;
    Table1FName6: TStringField;
    Table1LName6: TStringField;
    Table1Initial6: TStringField;
    Table1Institution6: TStringField;
    Table1Order6: TStringField;
    Query1Name: TStringField;
    procedure FormCreate(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dflag: string;
  dReviewerID: string;
  dbOption: string;
  i:integer;             {loop control variable}

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

    send ('<HTML><HEAD>');
    SendTitle('On-Line Article Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');

{Determine whether flag is valid }

dflag:= getsmallfield ('flag');
dReviewerID:= getsmallfield ('ReviewerID');

if (dflag = '1') then begin    {flag is set}

send('<center><H1>On-Line Article Review System</H1></center>');
sendhr;

    send('<p>In order to review an article, you may select');
    send('from one of the search options below:</p>');


  {Search by Keyword}

    send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/ArtSrch2.exe">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');                {send flag for verification}
    send('<INPUT TYPE="hidden" Name="Type" Value="keyword">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

    sendhr;
    sendhdr ('3', 'Choose from a list of Articles with the following keyword:');

        {Pull Keywords from KEYWORD Table and display on pull-down menu}
        with Query1 do begin
          close;
          sql.clear;
          sql.add('Select * FROM Keyword ');
          open;

          {Move records from query result to stringlist}
          Table3.open;
          Table3.first;

          send('<P><CENTER><B>Keyword: </B><SELECT NAME="Keyword">');
          send('<OPTION> Do not search on this field');

          while not Table3.EOF do begin
            dbOption:= Table3.fieldByName('Name').Asstring;
            send('<OPTION> ' +dbOption+ '');
            Table3.next;
            end; {for all records in the query result}
           send('</SELECT></CENTER>');
          end; {withQuery1}

     send('<P><CENTER><INPUT TYPE="submit" Value="Show me a list of Articles like this"> ');
     send('<INPUT TYPE="reset" VALUE="Reset Field"></CENTER></form>');

    {Search by Author Name}
     send('<P><FORM ACTION="../cgi-win/ArtSrch2.exe" METHOD="POST">');
     send('<INPUT TYPE="hidden" Name="flag" Value="1">');      {send flag for verification}
    send('<INPUT TYPE="hidden" Name="Type" Value="author">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

    sendhr;
    sendhdr ('3', 'Choose from a list of Articles by this Author?');
    send ('<P><CENTER><B>Author First Name:  </B><INPUT NAME="fname" Size="25"
TYPE="text">');
    send ('<P><B>Author Last Name:   </B><INPUT NAME="lname" Size="25" TYPE="text">');

    send('<P><INPUT TYPE="submit" Value="Find an Article by this Author"> ');
    send('<INPUT TYPE="reset" VALUE="Clear Names"></CENTER></form>');

   {Display Article with this Paper Number}

    send('<P><FORM ACTION="../cgi-win/ArtSrch2.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="Type" Value="number">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

    sendhr;
    sendhdr ('3', 'Enter the Paper Number of the article of interest:');

    send ('<CENTER><B>Paper Number: </B><INPUT NAME="Paper_Number" TYPE=TEXT>');
    send('<P><INPUT TYPE="submit" Value="Show me this this article."> ');
    send('<INPUT TYPE="reset" VALUE="Reset Field"></CENTER></form>');

   {Display ALL Paper Titles}

    send('<P><FORM ACTION="../cgi-win/ArtSrch2.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="Type" Value="all">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

    sendhr;
    sendhdr ('3', 'Choose from a list of ALL Article Submissions?');
    send('<P><CENTER><INPUT TYPE="submit" Value="Show me a list of ALL of the Paper
Titles."></CENTER></FORM>');


    sendhr;
    send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review Options
Page"></CENTER></FORM>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );

end  {flag is set}

else begin    {flag **incorrect**}

send ('<center><H2>On-Line Article Review System</H2></center>');
sendhr;
```

```
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');

send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

        end;  {flag **incorrect**}

    end; {with cgiEnvData1 do}

  end;   {FormCreate}

end.  {Application}
```

The Article Review Search Result page: **Artsrch2.exe**

```
unit Ar_srch2;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   Query2: TQuery;
   DataSource2: TDataSource;
   CGIDB2: TCGIDB;
   CGIDB3: TCGIDB;
   Query3: TQuery;
   Table4: TTable;
   DataSource3: TDataSource;
   Table4ANumber: TIntegerField;
   Table4LName: TStringField;
   Table4FName: TStringField;
   CGIDB1: TCGIDB;
   DataSource1: TDataSource;
   Query1: TQuery;
   Table2: TTable;
   Table3: TTable;
   Table3ANumber: TIntegerField;
   Table3LName: TStringField;
   Table3FName: TStringField;
   Table2ANumber: TIntegerField;
   Table2LName: TStringField;
   Table2FName: TStringField;
   Table1: TTable;
   Table1Paper_Number: TIntegerField;
   Table1Title: TStringField;
   Table1Invited: TStringField;
   Table1Accepted: TStringField;
   Table1ContactAuthorNumber: TIntegerField;
   Table1Session: TStringField;
   Table1OrderInSession: TIntegerField;
   Table1PresentationTime: TStringField;
   Table1Keyword1: TStringField;
   Table1Keyword2: TStringField;
   Table1Keyword3: TStringField;
   Table1Abstract: TMemoField;
   Table1ContactOrder: TStringField;
   Table1FName2: TStringField;
   Table1LName2: TStringField;
   Table1Initial2: TStringField;
   Table1Institution2: TStringField;
   Table1Order2: TStringField;
   Table1FName3: TStringField;
   Table1LName3: TStringField;
```

```
    Table1Initial3: TStringField;
    Table1Institution3: TStringField;
    Table1Order3: TStringField;
    Table1FName4: TStringField;
    Table1LName4: TStringField;
    Table1Initial4: TStringField;
    Table1Institution4: TStringField;
    Table1Order4: TStringField;
    Table1FName5: TStringField;
    Table1LName5: TStringField;
    Table1Initial5: TStringField;
    Table1Institution5: TStringField;
    Table1Order5: TStringField;
    Table1FName6: TStringField;
    Table1LName6: TStringField;
    Table1Initial6: TStringField;
    Table1Institution6: TStringField;
    Table1Order6: TStringField;
    Table1Year: TStringField;
    CGIDB4: TCGIDB;
    Query4: TQuery;
    DataSource4: TDataSource;
    Table5: TTable;
    Query4ANumber: TIntegerField;
    Query4FName: TStringField;
    Query4LName: TStringField;
    Table5Paper_Number: TIntegerField;
    Table5Title: TStringField;
    Table5ContactAuthorNumber: TIntegerField;
    procedure FormCreate(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;
  function MixCaseStr(S: string): string;

implementation

{$R *.DFM}

{****************************************************}
function MixCaseStr(S: string): string;
{ This function converts a string to Mixed case.
That is Capital first letter and the rest lower case. }
var
   i: integer;

begin

  if (S[1] >= 'a') and (S[1] <= 'z') then
```

```
 Dec(S[1], 32);
 for i := 2 to Length(S) do
 if (S[i] >= 'A') and (S[i] <= 'Z') and (S[i-1]<>' ') and (S[i-1]<>'e') and (S[i-1]<>'-') and (S[i-1]<>'c') then
 Inc(S[i], 32);
 MixCaseStr := S;

end;
{****************************************************}


{****************************************************}

procedure TForm1.FormCreate(Sender: TObject);
var
   i, count: integer;            {loop variable}
   temp:string;
   countstr: string;
   dPaper_Number: string;
   dANumber: string;
   dTitle: string;
   dflag: string;
   dReviewerID: string;
   dtype: string;
   dfnamein: string;
   dlnamein: string;
   dFName: string;
   dLName: string;
   dHonorific: string;
   dInstitution: string;
   dfirst: string;
   dlast: string;
   dContactAuthorNumber: string;

   dInstitution2: string;
   dkeyword: string;
   MySelector: integer;
   Selector: integer;
   TC: TDataSet;

begin
   with CGIEnvData1 do begin
      {Standard Header}
      webSiteINIFilename:=paramstr(1);
      application.onException:=cgiErrorHandler;
      application.processMessages;
      createStdout;
      sendPrologue;

      {HTML Header}
      send ('<HTML><HEAD>');
      SendTitle('On-Line Article Review System');
      send ('</HEAD><BODY bgcolor=FFFFFF>');

 {Determine whether flag is valid}
 dflag:= getsmallfield ('flag');
```

```
{Get ReviewerID}
dReviewerID:= getsmallfield ('ReviewerID');

if (dflag = '1') then begin    {flag is set}

send('<center><H1>On-Line Article Review System</H1></center>');
sendhr;

{Get type from search page}
dtype:= getsmallfield ('type');   {Flag to determine type of search}

{Decide what action is required}

if dtype = 'keyword' then  MySelector := 1;
if dtype = 'author' then  MySelector := 2;
if dtype = 'number' then  MySelector := 3;
if dtype = 'all' then  MySelector := 4;

case MySelector of

    1 : begin  {Keyword}

        send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/ArtView2.exe">');
        send('<INPUT TYPE="hidden" Name="flag" Value="1">');
        send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

        dkeyword:= getsmallfield ('keyword');

        if  dkeyword = 'Do not search on this field'
         then begin

         sendhdr('2','<CENTER>You must first select a keyword!</CENTER>');
         send('</FORM>');

        end  {if nothing selected}

        else begin   {Keyword}

        {Find the list of Titles based on keyword search}
        with Query2 do begin
            close;
            sql.clear;
            sql.add('Select * FROM Submissi WHERE ');

            {Keyword}
            if (dkeyword <> 'Do not search on this field') and (sql.count >= 2) then
               sql.add (' and ');       {need an 'and' between statements}

            if (dkeyword <> 'Do not search on this field') then
            sql.add('(Keyword1="'+dkeyword+'" or Keyword2="'+dkeyword+'" or
Keyword3="'+dkeyword+'")');
            sql.add( 'Order by Title' );
            open;
            count:= Recordcount;
```

```
          countstr:= inttostr (count);

          if recordcount = 0 then begin
            sendhdr('2', 'There were no records that matched your query!');
            send('</FORM>');
          end

          else begin

          send('<CENTER>There are '+countstr+' Titles that match your request.  Select the ');
          send('article that you would like to review and press submit.</p>');

          {Build list of Titles}
          send('<TABLE BORDER>');
          send('<TR><TH>Paper #</TH><TH>Title</TH><TH>Author</TH></TR>');

          While not Query2.EOF do begin
               dTitle:= fieldbyname('Title').asstring;
               dPaper_Number:= fieldbyname('Paper_Number').asstring;
               dANumber:= fieldbyname('ContactAuthorNumber').asstring;

               {Get Author Name information from the AUTHOR Table}
               with Table3 do begin
                    {Move to proper record}
                    open;
                    first;
                    while fieldbyName('ANumber').asstring <> dANumber do
                         next;
                         {Retrieve Author Name information}
                         dFName := fieldByName('FName').AsString;
                         dLName := fieldByName('LName').AsString;
                    close;
                    end; {with Table3}

        send ('<TR><TD><INPUT TYPE="radio" NAME="Paper_Number"
Value="'+dPaper_Number+'"><B> '+dPaper_Number+'</B></TD>');
               send ('<TD>'+dTitle+'</TD><TD>'+dFName+' '+dLName+'</TD></TR>');
               next;
               end;  {while not EOF}

         send('</TABLE></CENTER>');
         close;

         send('<P><CENTER><INPUT TYPE="submit" Value="Select this Paper for
Review"></CENTER></FORM>');

          end;  {else}
        end;  {keyword}
      end;  {query2}
    end;  {case Keyword}

    2: begin   {author}

       send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/ArtView2.exe">');
```

```
 send('<INPUT TYPE="hidden" Name="flag" Value="1">');
 send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

 {Get Author Name}
 dFNamein:= getsmallfield ('fname');
 dfname:= MixCaseStr(dFNamein);

 dLNamein:= getsmallfield ('lname');
 dlname:= MixCaseStr(dLNamein);

{Find out if name is in contact authors. Only the last name is required,
but the first name will help narrow the search. }

if (dlname <> '') then begin

  {Search for name in AUTHOR Table}
  with Query4 do begin
    close;
    sql.clear;

    {If both the first and last name is given, search for match on both}
    if (dfname <> '') then
    sql.add('Select * FROM Author WHERE FName="'+dfname+'" and LName="'+dlname+'"')

    {Only the last name is given, so search only on it}
    else
    sql.add('Select * FROM Author WHERE LName="'+dlname+'"');

    {execute the Query}
    open;

    {Display results}
    send ('<P><center>');
    If Recordcount > 0 then begin

      send('<CENTER>Here is the article(s) by the author you requested.<P>');
      send('<CENTER><TABLE BORDER>');
      send('<TR><TH>Paper #</TH><TH>Title</TH><TH>Author</TH></TR>');

      While not Query4.EOF do begin

        dFName := fieldByName('FName').AsString;
        dLName := fieldByName('LName').AsString;
        dContactAuthorNumber:= fieldbyname('ANumber').asstring;

        {Get Author Name information from the AUTHOR Table}
        with Table5 do begin
            {Move to proper record}
            open;
            first;
            while fieldbyName('ContactAuthorNumber').asstring <> dContactAuthorNumber do
                next;
                {Retrieve Author Name information}
                dTitle := fieldByName('Title').AsString;
```

```
                    dPaper_Number := fieldByName('Paper_Number').AsString;
                    end; {with Table5}

    send ('<TR><TD><INPUT TYPE="radio" NAME="Paper_Number"
Value="'+dPaper_Number+'"><B> '+dPaper_Number+'</B></TD>');
                send ('<TD>'+dTitle+'</TD><TD>'+dFName+' '+dLName+'</TD></TR>');
                next;
                end;  {while not EOF}

                send('</TABLE></CENTER>');
                close;

                send('<P><CENTER><INPUT TYPE="submit" Value="Select this Paper for
Review"></CENTER></FORM>');

            end

            else begin  {The name is not a Contact Author.}

    sendhdr('2', 'There are no authors in the database with the name <B>' + dfname + ' ' + dlname + '</B>.');
            send('</FORM>');

            end;    {if recordcount > 0}

          end;  {withQuery4}

        end    {If dlname <>'' then}

        else begin

        sendhdr('2','<CENTER>You must enter a last name!</CENTER>');
        send('</FORM>');

        end;  {If dlname <>'' else}

    end;   {case author}


    3 : begin  {Paper Number search}

      send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/ArtView2.exe">');
      send('<INPUT TYPE="hidden" Name="flag" Value="1">');
      send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

      dPaper_Number:= getsmallfield ('Paper_Number');

      {Get Paper Information from Submissi Table}
      with Query3 do begin
         close;
         sql.clear;
         sql.add('Select * FROM Submissi');
         sql.add('Where Paper_Number = '+dPaper_Number+'');
         open;

         if recordcount = 0 then begin
```

```
            sendhdr('2','<CENTER>There Are no records that match your request!</CENTER>');
            send('</FORM>');
        end

        else begin

        send('<CENTER>Here is the article that matches your request.<P>');
        send('<CENTER><TABLE BORDER>');
        send('<TR><TH>Paper #</TH><TH>Title</TH><TH>Author</TH></TR>');

        While not Query3.EOF do begin

            {Build the check boxes of Titles}
            dTitle:= fieldbyname('Title').asstring;
            dPaper_Number:= fieldbyname('Paper_Number').asstring;
            dANumber:= fieldbyname('ContactAuthorNumber').asstring;

            {Get Author Name information from the AUTHOR Table}
            with Table4 do begin
                {Move to proper record}
                open;
                first;
                while fieldbyName('ANumber').asstring <> dANumber do
                    next;
                    {Retrieve Author Name information}
                    dFName := fieldByName('FName').AsString;
                    dLName := fieldByName('LName').AsString;
                    end; {with Table4}

    send ('<TR><TD><INPUT TYPE="radio" NAME="Paper_Number"
Value="'+dPaper_Number+'"><B> '+dPaper_Number+'</B></TD>');
            send ('<TD>'+dTitle+'</TD><TD>'+dFName+' '+dLName+'</TD></TR>');
            next;
            end;  {while not EOF}

            send('</TABLE></CENTER>');
            close;

send('<P><CENTER><INPUT TYPE="submit" Value="Select this Paper for
   Review"></CENTER></FORM>');

            end; {if}
        end;  {with Query3}
      end; {Paper Number Search}

   4 :  begin  {ALL}

       send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/Artview2.exe">');
       send('<INPUT TYPE="hidden" Name="flag" Value="1">');
       send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');

       with Query1 do begin

          {prepare;    { Optimizes query }
          close;
          sql.clear;
          sql.add('Select * FROM Submissi');
          sql.add('Order by Paper_Number');
          open;
          count:= recordcount;
          countstr:= inttostr (count);

          if recordcount = 0 then begin
            sendhdr('2', '<CENTER>There are no records in the database.</CENTER>');
            send('</FORM>');
          end

          else begin

          send('<P>There are '+countstr+' articles in the database.  Please select the radio button');
          send('of the article you would like to view and then press the selection button below the
table.<P>');
          {Build list of Titles}
          first;
          send('<CENTER><TABLE BORDER>');
          send('<TR><TH>Paper #</TH><TH>Title</TH><TH>Author</TH></TR>');

          While not Query1.EOF do begin

              dTitle:= fieldbyname('Title').asstring;
              dPaper_Number:= fieldbyname('Paper_Number').asstring;
              dANumber:= fieldbyname('ContactAuthorNumber').asstring;

              {Get Author Name information from the AUTHOR Table}
              with Table2 do begin

                  {Move to proper record}
                  open;
                  first;
                  while fieldbyName('ANumber').asstring <> dANumber do
                      next;

                      {Retrieve Author Name information}
                      dFName := fieldByName('FName').AsString;
                      dLName := fieldByName('LName').AsString;
                  close;
                  end; {with Table2}

        send ('<TR><TD><INPUT TYPE="radio" NAME="Paper_Number"
Value="'+dPaper_Number+'"><B> '+dPaper_Number+'</B></TD>');
              send ('<TD>'+dTitle+'</TD><TD>'+dFName+' '+dLName+'</TD></TR>');
              next;
              end;  {while not EOF}

              send('</TABLE>');
              close;

              send('<P><INPUT TYPE="submit" Value="Select a Paper for
Review"><P></CENTER></FORM>');
              {send('<INPUT TYPE="reset" VALUE="Reset values"></CENTER></FORM>');}
            end;    {else}
          end;   {with Query1}
        end;   {if all}

  end; {case}

        {HTML Footer}
        sendhr;
        send('<P><FORM ACTION="../cgi-win/Artrev.exe" METHOD="POST">');
        send('<INPUT TYPE="hidden" Name="flag" Value="1">');
        send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
        send('<P><CENTER><INPUT TYPE="submit" Value="Query another
Article"></CENTER></FORM>');

        sendhr;
        send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
        send('<INPUT TYPE="hidden" Name="flag" Value="1">');
        send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
        send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review
Options Page"></CENTER></FORM>');
        send ('</BODY></HTML>');
        closeStdout;
        closeApp( application );
end  {flag is set}

else begin    {flag **incorrect**}

send ('<center><H2>On-Line Article Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');

send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('Return to the <A HREF="http://xxx.xxx.xxx.xxx/review/Review30.html">On-line Article Review
Page</A>');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

      end; {flag **incorrect**}

    end; {with cgiEnvData1 do}

  end;  {Procedure TForm1.FormCreate}

end.
```

The Article Review and Vote page: **Artview2.exe**

```
unit Ar_view2;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
  TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  Table2: TTable;
  Table3: TTable;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);

 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

 theabstract : TStringList;
 theAuthors : TStringList;
 i, counter: integer;
 dflag:  string;

   dbContactOrderInt: integer;
   dbOrder2int: integer;
   dbOrder3int: integer;
   dbOrder4int: integer;
   dbOrder5int: integer;
   dbOrder6int: integer;

   dPaper_Number: string;
   dbOption: string;
   dTitle: string;
   dbANumber: string;
   dbabstract: string;
   dbcontactOrder: string;

   dbfname2: string;
   dblname2: string;
   dbinitial2: string;
   dbinstitution2: string;
   dborder2: string;
```

```
    dbfname3: string;
    dblname3: string;
    dbinitial3: string;
    dbinstitution3: string;
    dborder3: string;

    dbfname4: string;
    dblname4: string;
    dbinitial4: string;
    dbinstitution4: string;
    dborder4: string;

    dbfname5: string;
    dblname5: string;
    dbinitial5: string;
    dbinstitution5: string;
    dborder5: string;

    dbfname6: string;
    dblname6: string;
    dbinitial6: string;
    dbinstitution6: string;
    dborder6: string;

    dbFName : string;
    dbLName : string;
    dbHonorific : string;
    dbInitial: string;
    dbInstitution : string;

    dReviewerID:  string;
    dVote:  string;
    dRNumber:  string;
    dReview: string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
      with CGIEnvData1 do
        begin

          webSiteINIFilename:=paramstr(1);
          application.onException:=cgiErrorHandler;
          application.processMessages;

          createStdout;
          sendPrologue;

          {Send the html page}
          send ('<HTML><HEAD>');
          SendTitle('On-Line Article Review System');
          send ('</HEAD><BODY bgcolor=FFFFFF>');
        send ('<center><H1>Article Review</H1></center><HR>');

    {Get the Paper Number Selected}
    dPaper_Number:= getsmallfield('Paper_Number');
    dReviewerID:= getsmallfield('ReviewerID');

    {Determine whether flag is valid}
    dflag:= getsmallfield ('flag');

    if (dflag = '1') then begin    {flag is set}

    counter := 0;

    with Query1 do         {Check the database for Reviewer entry for this submission}
    begin
        close;
        sql.clear;
        sql.add('Select * FROM Reviews WHERE Paper_Number='''+dPaper_Number+'''' + 'AND
ReviewerID='''+dReviewerID+'''');
        open;

        counter := Recordcount;

    if counter = 0 then begin          {No records returned}

      {Get the Paper Record from the SUBMISSION Table}
      with Table1 do begin

        {Move to proper record}
        open;
        first;
        while fieldbyName('Paper_Number').asstring <> dPaper_Number do
            next;

        {Retrieve record}
        dTitle:= fieldByName ('Title').asstring;
        dbANumber:= fieldByName ('ContactAuthorNumber').asstring;

        CGIDB1.memoToStringList (fieldbyname ('Abstract'), theabstract);

        dbcontactOrder:= fieldByName ('contactOrder').asstring;

        dborder2:= fieldByName ('order2').asstring;
        dbfname2:= fieldByName ('fname2').asstring;
        dblname2:= fieldByName ('lname2').asstring;
        dbinstitution2:= fieldByName ('institution2').asstring;

        dborder3:= fieldByName ('order3').asstring;
        dbfname3:= fieldByName ('fname3').asstring;
        dblname3:= fieldByName ('lname3').asstring;
        dbinstitution3:= fieldByName ('institution3').asstring;

        dborder4:= fieldByName ('order4').asstring;
        dbfname4:= fieldByName ('fname4').asstring;
        dblname4:= fieldByName ('lname4').asstring;
```

```
 dbinstitution4:= fieldByName ('institution4').asstring;

 dborder5:= fieldByName ('order5').asstring;
 dbfname5:= fieldByName ('fname5').asstring;
 dblname5:= fieldByName ('lname5').asstring;
 dbinstitution5:= fieldByName ('institution5').asstring;

 dborder6:= fieldByName ('order6').asstring;
 dbfname6:= fieldByName ('fname6').asstring;
 dblname6:= fieldByName ('lname6').asstring;
 dbinstitution6:= fieldByName ('institution6').asstring;
 close;
 end;


{Get Author Name information from the AUTHOR Table}
with Table2 do begin

 {Move to proper record}
 open;
 first;
 while fieldbyName('ANumber').asstring <> dbANumber do
     next;

 {Retrieve Author Name information}
 dbFName := fieldByName('FName').AsString;
 dbLName := fieldByName('LName').AsString;
 dbInstitution := fieldByName('Institution').AsString;

 {Determine Order of Authors}
 theAuthors:= TStringList.create;

 {Send the Authors}
 send ('<B>Author(s): </B><BR>');

 if dbContactOrder = '' then
   send (' <I>'+ dbFName +' '+ dbLName +', '+dbInstitution+'</I> ')
   else begin

   {Initialize the stringlist}
   for i:= 0 to 6 do
      theAuthors.add('not used');

      dbContactOrderInt:= strtoint(dbContactOrder);
      theAuthors[dbContactOrderInt]:= '' +dbFName+ ' ' +dbLName+ ', '+dbInstitution+'';

      if dbOrder2 <> '' then begin
        dbOrder2Int:= strtoint(dbOrder2);
        theAuthors[dbOrder2Int]:= ''+dbFName2+ ' ' +dbLName2+ ', '+dbInstitution2+'';
        end;

      if dbOrder3 <> '' then begin
        dbOrder3Int:= strtoint(dbOrder3);
        theAuthors[dbOrder3Int]:= ''+dbFName3+ ' ' +dbLName3+ ', '+dbInstitution3+'';
        end;
```

```
            if dbOrder4 <> '' then begin
              dbOrder4Int:= strtoint(dbOrder4);
              theAuthors[dbOrder4Int]:= ''+dbFName4+ ' ' +dbLName4+ ', '+dbInstitution4+'';
              end;

            if dbOrder5 <> '' then begin
              dbOrder5Int:= strtoint(dbOrder5);
              theAuthors[dbOrder5Int]:= ''+dbFName5+ ' ' +dbLName5+ ', '+dbInstitution5+'';
              end;

            if dbOrder6 <> '' then begin
              dbOrder6Int:= strtoint(dbOrder6);
              theAuthors[dbOrder6Int]:= ''+dbFName6+ ' ' +dbLName6+ ', '+dbInstitution6+'';
              end;

          send ('<I>' +theAuthors[1]+ '</I><BR>');

          for i:= 2 to 6 do
            if theAuthors[i] <> 'not used' then send ('<I>' +theAuthors[i]+ '</I><BR>');

            end; {else}

        close;
        end;  {with Table2}

     send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
     send ('<H3><B>Title:  </B>  " ' + dTitle + ' " </H3>');
     send ('<B>Paper Number:    </B>   " ' + dPaper_Number + ' " ');
     send ('<P><B>Abstract:    </B><P>');
     for i := 0 to theabstract.count - 1 do
       send( theabstract.strings[i] );
     theabstract.free;  {release the memory held by 'theabstract'}

     sendhr;
     send ('<P>You may view the extended summary for this submission by selecting');
     send ('file '+dPaper_Number+' from the<A HREF="../uploads/"> following list<A
HREF="../uploads/"></A>.<P>');

    {Reviewer Input}
     sendhr;
     send ('<I>Please cast your vote for this particular article,</I><B> '+dReviewerID+':</B>');
     send('<P><FORM METHOD="POST" ACTION="../cgi-win/add133.exe" >');
     send ('<INPUT TYPE="hidden" Name="flag" Value="'+ dflag +'">');
     send ('<INPUT TYPE="hidden" Name="Title" Value="'+ dTitle +'">');
     send ('<INPUT TYPE="hidden" Name="Paper_Number" Value="'+ dPaper_Number +'">');
     send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');

     send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Strong Accept" CHECKED> Strong Accept');
     send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Conditional Accept" > Conditional Accept');
     send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Reject" > Reject');
     send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Undecided" > Undecided');
     send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Not Qualified" > Not Qualified');
     send ('<P><B>Review comments:    </B><P>');
     send('<TEXTAREA NAME="review" COLS=85 Rows=5 wrap=virtual>');
    send('Enter review comments here.  Please limit the length of your review to 100
words.</TEXTAREA>');
    send(' ');
    send ('<BR>');

    send('<P><CENTER><INPUT TYPE="submit" Value="Submit Review
Info"></CENTER></FORM>');

    end

    else if counter <> 0 then begin     {Only one record returned}

      send('<CENTER>You have already reviewed this particular article.</CENTER>');

    end; {if then else if}

  end; {with Query1}

    sendhr;
    send('<P><FORM ACTION="../cgi-win/ArtRev.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Review Another
Article"></CENTER></FORM>');

    sendhr;
    send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review Options
page"></CENTER></FORM>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );      { don't leave form around }

end  {flag is set}

else begin     {flag **incorrect**}

send ('</HEAD><BODY bgcolor=FFFFFF>');
send ('<center><H2>On-Line Article Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');


send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('Return to the <A HREF="http://XXX.XXX.XXX.XXX/review/Review30.html">On-line Article
Review Page</A>');
send ('</BODY></HTML>');
closeStdout;
```

```
closeApp( application );

       end;    {flag **incorrect**}
     end;    {with CGIEnvData1}
  end;    {with FormCreate}
end.
```

The review confirmation page: **Add133.exe**

```
unit D133add;

 {WHEN CALLED: This program adds article review data to the Reviews database.

 INPUT: Review data from the Artview2 page.

 ACTION: places Review data into the Reviews Table.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   Table1: TTable;
   DataSource1: TDataSource;
   Query1: TQuery;

   procedure FormCreate(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
  Form1: TForm1;
  counter: integer;
  thereview : TStringList;
  theabstract : TStringList;
  dstorabs : TStringList;
  dRNumber: string;
  dReviewerID: string;
  dVote:  string;
  dbTitle:  string;
  dPaper_Number: string;
  dbAbstract: string;
  dYear: string;
  dflag:  string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

    with CGIEnvData1 do begin
 {Standard Header}
 webSiteINIFilename:=paramstr(1);
 application.onException:=cgiErrorHandler;
 application.processMessages;

 createStdout;
 sendPrologue;

 {Get data from FORM}
 dReviewerID:= getsmallfield ('ReviewerID');
 dVote:= getsmallfield ('Vote');
 dbTitle:= getsmallfield ('Title');
 dPaper_Number:= getsmallfield ('Paper_Number');
 dYear := DateToStr(Now);

 thereview := TStringList.create;
 CGIEnvData1.getTextArea('review', thereview);

 {Determine whether flag is valid }
 dflag:= getsmallfield ('flag');

 if (dflag = '1') then begin    {flag is set}

   counter := 0;

 with Query1 do        {Check the database for Reviewer entry for this submission}
   begin
   close;
   sql.clear;
   sql.add('Select * FROM Reviews WHERE Paper_Number='''+dPaper_Number+'''' + 'AND
ReviewerID='''+dReviewerID+'''');
   open;

   counter := Recordcount;

  if counter = 0 then begin        {No records returned}

  {Place Data into Reviews Table}
   with Table1 do
     begin

        {Append new submission data to Reviews Table}
        open;
        AppendRecord([nil, dReviewerID, dPaper_Number, dbTitle, dVote, nil,  dYear]);

        edit;
        CGIDB1.StringlistToMemo(thereview, fieldbyname('Review'));
        thereview.free;

        last;
        dRNumber:=Table1.fieldbyname('RNumber').asstring;
        close;

     end;  {with Reviews}
 {Send the Thank you Paper page with the link to the On-Line Article Review Page}
```

```
   send ('<HTML><HEAD>');
   SendTitle('On-Line Article Review System');

   send ('</HEAD><BODY bgcolor=FFFFFF>');
   send ('<center><H1>Article Review</H1></center>');

   sendhr;
   send ('<P>Thank you for your input,<B> '+ dReviewerID +'</B>. ');

   send ('Your information has been added to the article review database.</p>');

   end

   else if counter <> 0 then begin     {Only one record returned}

   {Send the Thank you Paper page with the link to the On-Line Article Review Page}
   send ('<HTML><HEAD>');
   SendTitle('On-Line Article Review System');

   send ('</HEAD><BODY bgcolor=FFFFFF>');
   send ('<center><H1>Article Review</H1></center>');
   sendhr;

   send ('You have already reviewed this particular article.');

   end; {if then else if}

   end; {with Query1}

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Artrev.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Query another
Article"></CENTER></FORM>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review Options
Page"></CENTER></FORM>');

   send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
   send ('</BODY></HTML>');
   closeStdout;
   closeApp( application );    { don't leave form around }

  end  {flag is set}

  else begin     {flag **incorrect**}

  send ('</HEAD><BODY bgcolor=FFFFFF>');
  send ('<center><H2>On-line Article Review System</H2></center>');
  sendhr;
  send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
  send ('<P>Please ensure that you are authorized to access this information.');
  send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login
again.</A>');
  send ('<P>If you are not authorized to access this information, please note that a log is');
  send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
  send ('prosecution for your actions.<BR>');

  send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
  send ('Return to the <A HREF="../review/Review30.htm">On-line Article Review Log-on Page</A>');
  send ('<BR>');
  send ('</BODY></HTML>');
  closeStdout;
  closeApp( application );

        end;    {flag **incorrect**}

     end;    {with CGIEnvData1}

   end;    {FormCreate}

end.
```

The edit a review page: **Editrev.exe**

```
unit Edit_rev;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  Table1: TTable;
  Query1: TQuery;
  DataSource1: TDataSource;
  Table1ReviewerID: TStringField;
  Table1Vote: TStringField;
  Table1Paper_Number: TStringField;
  Table1Title: TStringField;
  Query1Title: TStringField;
  Table1Review: TMemoField;
  Table1RNumber: TIntegerField;
  Query1Paper_Number: TStringField;
  Table2: TTable;
  Table3: TTable;
  Table2Paper_Number: TIntegerField;
  Table2ContactAuthorNumber: TIntegerField;
  Table3ANumber: TIntegerField;
  Table3LName: TStringField;
  Table3FName: TStringField;
  procedure FormCreate(Sender: TObject);
  {procedure CGIDB1SendingHotField(currentRecord: TDataset;
   var s: OpenString);}


 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;


implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var

   i, count: integer;          {loop variable}
```

```
    temp:string;
    countstr: string;
    dPaper_Number: string;
    dflag: string;
    dReviewerID: string;
    dRNumber: string;
    dTitle: string;
    dVote: string;
    dbFName: string;
    dbLName: string;
    dANumber: string;

begin
    with CGIEnvData1 do begin
        {Standard Header}
        webSiteINIFilename:=paramstr(1);
        application.onException:=cgiErrorHandler;
        application.processMessages;
        createStdout;
        sendPrologue;

        {HTML Header}
        send ('<HTML><HEAD>');
        SendTitle('On-Line Article Review System');
        send ('</HEAD><BODY bgcolor=FFFFFF>');

{Determine whether flag is valid}
dflag:= getsmallfield ('flag');
dReviewerID:= getsmallfield ('ReviewerID');

if (dflag = '1') then begin    {flag is set}

send('<center><H1>Edit a Review</H1></center>');
sendhr;

send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/Revview.exe">');
send('<INPUT TYPE="hidden" Name="flag" Value="1">');
send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');


    with Query1 do begin
      prepare;    { Optimizes query }
      close;
      sql.clear;
      sql.add('SELECT * FROM Reviews');
      sql.add('WHERE ReviewerID = "'+dReviewerID+'"');
      sql.add('Order by Paper_Number');
      open;
      count:= Recordcount;
      countstr:= inttostr (count);
      send('<CENTER><P>You have ' +countstr+ ' reviews in the database.<P>');
      first;

      send('<TABLE BORDER>');
      send('<TR><TH>Paper #</TH><TH>Title</TH><TH>Author</TH></TR>');
```

```
While not Query1.EOF do begin

 {Build the check boxes of Titles}
 dTitle:= fieldbyname('Title').asstring;
 dPaper_Number:= fieldbyname('Paper_Number').asstring;

 {Get ContactAuthorNumber from the Submissi Table}
 with Table2 do begin

    {Move to proper record}
    open;
    first;
    while fieldbyName('Paper_Number').asstring <> dPaper_Number do
        next;

    {Retrieve Author Number}
    dANumber:= fieldbyname('ContactAuthorNumber').asstring;
    end; {with Table2}

 {Get Author Name information from the AUTHOR Table}
 with Table3 do begin

    {Move to proper record}
    open;
    first;
    while fieldbyName('ANumber').asstring <> dANumber do
        next;

    {Retrieve Author Name information}
    dbFName := fieldByName('FName').AsString;
    dbLName := fieldByName('LName').AsString;

    end; {with Table3}

 send('<INPUT TYPE="hidden" Name="RNumber" Value="'+dRNumber+'">');
 send ('<TR><TD><INPUT TYPE="radio" NAME="Paper_Number"
Value="'+dPaper_Number+'"><B> '+dPaper_Number+'</B></TD>');
 send ('<TD>'+dTitle+'</TD><TD>'+dbFName+' '+dbLName+'</TD></TR>');
 next;
 end; {while not EOF}

 send('</TABLE></CENTER>');
 send('<P>');
 close;

send('<P><CENTER><INPUT TYPE="submit" Value="Select this Paper for Review"><P>');
send('<INPUT TYPE="reset" VALUE="Reset values"></CENTER></FORM>');
end; { with Query1}

 {HTML Footer}
 sendhr;
 send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
 send('<INPUT TYPE="hidden" Name="flag" Value="1">');
 send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
```

```
        send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review
Options Page"></CENTER></FORM>');
        send ('</BODY></HTML>');
        closeStdout;
        closeApp( application );

end  {flag is set}

else begin     {flag **incorrect**}

send ('<center><H2>On-Line Article Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');


send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

          end;   {flag **incorrect**}

       end;   {with cgiEnvData1 do}

    end;  {Procedure TForm1.FormCreate}

end.
```

The edit a review page: **Revview.exe**

```
unit Re_view;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
   TForm1 = class(TForm)
   DataSource1: TDataSource;
   Table1: TTable;
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   Table2: TTable;
   Table3: TTable;
   Query1: TQuery;
   procedure FormCreate(Sender: TObject);

  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  Form1: TForm1;

  theabstract : TStringList;
  theAuthors : TStringList;
  thereview : TStringList;
  i: integer;

     dbContactOrderInt: integer;
     dbOrder2int: integer;
     dbOrder3int: integer;
     dbOrder4int: integer;
     dbOrder5int: integer;
     dbOrder6int: integer;

     dPaper_Number: string;
     dbOption: string;
     dTitle: string;
     dbANumber: string;
     dbabstract: string;
     dbcontactOrder: string;

     dbfname2: string;
     dblname2: string;
     dbinitial2: string;
     dbinstitution2: string;
     dborder2: string;
```

```
    dbfname3: string;
    dblname3: string;
    dbinitial3: string;
    dbinstitution3: string;
    dborder3: string;

    dbfname4: string;
    dblname4: string;
    dbinitial4: string;
    dbinstitution4: string;
    dborder4: string;

    dbfname5: string;
    dblname5: string;
    dbinitial5: string;
    dbinstitution5: string;
    dborder5: string;

    dbfname6: string;
    dblname6: string;
    dbinitial6: string;
    dbinstitution6: string;
    dborder6: string;

    dbFName : string;
    dbLName : string;
    dbHonorific : string;
    dbInitial: string;
    dbInstitution : string;

    dReviewerID:  string;
    dVote:  string;
    dRNumber:  string;
    dReview: string;
    dflag: string;


implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
     with CGIEnvData1 do
       begin

        webSiteINIFilename:=paramstr(1);
        application.onException:=cgiErrorHandler;
        application.processMessages;

        createStdout;
        sendPrologue;

        {Send the html page}
        send ('<HTML><HEAD>');
    SendTitle('On-Line Article Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');
    send ('<center><H1>Edit a Review</H1></center><HR>');

{Determine whether flag is valid}
dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

{Get the Paper Number Selected}
dPaper_Number:= getsmallfield('Paper_Number');
dReviewerID:= getsmallfield('ReviewerID');

{Get the Paper Record from the SUBMISSION Table}
with Table1 do begin

    {Move to proper record}
    open;
    first;
    while fieldbyName('Paper_Number').asstring <> dPaper_Number do
        next;

    {Retrieve record}
    dTitle:= fieldByName ('Title').asstring;
    dbANumber:= fieldByName ('ContactAuthorNumber').asstring;

    CGIDB1.memoToStringList (fieldbyname ('Abstract'), theabstract);

    dbcontactOrder:= fieldByName ('contactOrder').asstring;

    dborder2:= fieldByName ('order2').asstring;
    dbfname2:= fieldByName ('fname2').asstring;
    dblname2:= fieldByName ('lname2').asstring;
    dbinstitution2:= fieldByName ('institution2').asstring;

    dborder3:= fieldByName ('order3').asstring;
    dbfname3:= fieldByName ('fname3').asstring;
    dblname3:= fieldByName ('lname3').asstring;
    dbinstitution3:= fieldByName ('institution3').asstring;

    dborder4:= fieldByName ('order4').asstring;
    dbfname4:= fieldByName ('fname4').asstring;
    dblname4:= fieldByName ('lname4').asstring;
    dbinstitution4:= fieldByName ('institution4').asstring;

    dborder5:= fieldByName ('order5').asstring;
    dbfname5:= fieldByName ('fname5').asstring;
    dblname5:= fieldByName ('lname5').asstring;
    dbinstitution5:= fieldByName ('institution5').asstring;

    dborder6:= fieldByName ('order6').asstring;
    dbfname6:= fieldByName ('fname6').asstring;
    dblname6:= fieldByName ('lname6').asstring;
    dbinstitution6:= fieldByName ('institution6').asstring;
    close;
    end;

{Get Author Name information from the AUTHOR Table}
with Table2 do begin

    {Move to proper record}
    open;
    first;
    while fieldbyName('ANumber').asstring <> dbANumber do
        next;

    {Retrieve Author Name information}
    dbFName := fieldByName('FName').AsString;
    dbLName := fieldByName('LName').AsString;
    dbInstitution := fieldByName('Institution').AsString;

    {Determine Order of Authors}
    theAuthors:= TStringList.create;

    {Send the Authors}
    send ('<B>Author(s): </B><BR>');

    if dbContactOrder = '' then
      send (' <I>'+ dbFName +' '+ dbLName +', '+dbInstitution+'</I> ')
      else begin

      {Initialize the stringlist}
      for i:= 0 to 6 do
         theAuthors.add('not used');

         dbContactOrderInt:= strtoint(dbContactOrder);
         theAuthors[dbContactOrderInt]:= '' +dbFName+ ' ' +dbLName+ ', '+dbInstitution+'';

         if dbOrder2 <> '' then begin
           dbOrder2Int:= strtoint(dbOrder2);
           theAuthors[dbOrder2Int]:= ''+dbFName2+ ' ' +dbLName2+ ', '+dbInstitution2+'';
           end;

         if dbOrder3 <> '' then begin
           dbOrder3Int:= strtoint(dbOrder3);
           theAuthors[dbOrder3Int]:= ''+dbFName3+ ' ' +dbLName3+ ', '+dbInstitution3+'';
           end;

         if dbOrder4 <> '' then begin
           dbOrder4Int:= strtoint(dbOrder4);
           theAuthors[dbOrder4Int]:= ''+dbFName4+ ' ' +dbLName4+ ', '+dbInstitution4+'';
           end;

         if dbOrder5 <> '' then begin
           dbOrder5Int:= strtoint(dbOrder5);
           theAuthors[dbOrder5Int]:= ''+dbFName5+ ' ' +dbLName5+ ', '+dbInstitution5+'';
           end;

         if dbOrder6 <> '' then begin
           dbOrder6Int:= strtoint(dbOrder6);
```

```
            theAuthors[dbOrder6Int]:= ''+dbFName6+ ' ' +dbLName6+ ', '+dbInstitution6+'';
            end;

        send ('<I>' +theAuthors[1]+ '</I><BR>');

        for i:= 2 to 6 do
           if theAuthors[i] <> 'not used' then send ('<I>' +theAuthors[i]+ '</I><BR>');

           end; {else}

      close;
      end;  {with Table2}

   {Get the Review Information from the REVIEWS Table}
   with Table3 do begin

      with Query1 do begin
      close;
      sql.clear;
      sql.add('SELECT * FROM Reviews');
      sql.add('WHERE ReviewerID = '''+dReviewerID+''' AND');
      sql.add('Paper_Number = '''+dPaper_Number+'''');
      open;

      {Move to proper record}
      first;
      while fieldbyName('Paper_Number').asstring <> dPaper_Number do
            next;
      {Retrieve record}
      dTitle:= fieldByName('Title').asstring;
      dRNumber:= fieldByName('RNumber').asstring;
      dVote:= fieldByName('Vote').asstring;
      CGIDB1.memoToStringList (fieldbyname('Review'), thereview);
      end; {with query1}
      close;
   end; {with Table3}

   send ('<H3><B>Title:  </B>  " ' + dTitle + ' " </H3>');
   send ('<B>Paper Number:    </B>    " ' + dPaper_Number + ' " ');
   send ('<P><B>Abstract:    </B><P>');
   for i := 0 to theabstract.count - 1 do
      send( theabstract.strings[i] );
      theabstract.free;  {release the memory held by 'theabstract'}

   sendhr;
   send ('<P>You may view the extended summary for this submission by selecting');
   send ('file '+dPaper_Number+' from the<A HREF="../uploads/"> following list<A
HREF="../uploads/"></A>.<P>');

   {Reviewer Input}
   sendhr;
   send ('<I><B>Please edit your review for this particular article:</B></I>');
   send ('<P><FORM METHOD="POST" ACTION="../cgi-win/add133b.exe">');
   send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send ('<INPUT TYPE="hidden" Name="Title" Value="'+ dTitle +'">');
```

```
    send ('<INPUT TYPE="hidden" Name="Paper_Number" Value="'+ dPaper_Number +'">');
    send ('<INPUT TYPE="hidden" Name="RNumber" Value="'+ dRNumber +'">');
    send ('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');


    send ('<P><B>Original Vote:    </B>  " ' + dVote + ' " <P>');

    send('<P><B>New Vote:   </B><P>');
    send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Strong Accept" > Strong Accept');
    send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Conditional Accept" > Conditional Accept');
    send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Reject" > Reject');
    send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Undecided" > Undecided');
    send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Not Qualified" > Not Qualified');

    send ('<P><B>Original Review: </B> (Simply edit or leave as is.)<P><TEXTAREA NAME="review"
COLS=85 Rows=5 wrap=virtual>');
    for i := 0 to thereview.count - 1 do
      send( thereview.strings[i] );
    send('</TEXTAREA>');
    thereview.free;  {release the memory held by 'thereview'}

    send ('<BR>');

    send('<P><CENTER><INPUT TYPE="submit" Value="Submit Review
Info"></CENTER></FORM>');

    sendhr;
    send('<P><FORM ACTION="../cgi-win/Editrev.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Edit Another
Review"></CENTER></FORM>');

    sendhr;
    send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+dReviewerID+'">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review Options
page"></CENTER></FORM>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application ); { don't leave form around }

end  {flag is set}

else begin     {flag **incorrect**}

send ('</HEAD><BODY bgcolor=FFFFFF>');
send ('<center><H2>On-Line Article Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');


send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('Return to the <A HREF="http://XXX.XXX.XXX.XXX/review/Review30.html">On-line Article
Review Page</A>');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

        end;    {flag **incorrect**}
      end;   {with CGIEnvData1}
   end;  {with FormCreate}
end.
```

The edit a review confirmation page: **Add133b.exe**

```
unit D133addb;

{WHEN CALLED: This program adds article review data to the Reviews database.

 INPUT: Review data from the Revview2 page.

 ACTION: places Review data into the Reviews Table.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   Table1: TTable;
   DataSource1: TDataSource;


   procedure FormCreate(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
  Form1: TForm1;
  thereview : TStringList;
  dRNumber: string;
  dRNumberint: integer;
  dReviewerID: string;
  dVote:  string;
  dbTitle:  string;
  dPaper_Number: string;
  dbAbstract: string;
  dYear: string;
  dflag: string;


implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);
begin

   with CGIEnvData1 do begin
```

```
{Standard Header}
webSiteINIFilename:=paramstr(1);
application.onException:=cgiErrorHandler;
application.processMessages;

createStdout;
sendPrologue;

{Determine whether flag is valid }
dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

  {Get data from FORM}
  dReviewerID:= getsmallfield ('ReviewerID');
  dVote:= getsmallfield ('Vote');
  dbTitle:= getsmallfield ('Title');
  dPaper_Number:= getsmallfield ('Paper_Number');
  dRNumber:= getsmallfield ('RNumber');
  dYear := DateToStr(Now);

  thereview := TStringList.create;
  CGIEnvData1.getTextArea('review', thereview);

  {Place Data into Reviews Table}
  with Table1 do begin

   {Move to proper record}
   open;
   first;
   while fieldbyName('RNumber').asstring <> dRNumber do
        next;

   {Update record }
   edit;
   SetFields([nil, dReviewerID, dPaper_Number, dbTitle, dVote, nil,  dYear]);

   edit;
   CGIDB1.StringlistToMemo(thereview, fieldbyname('Review'));
   thereview.free;

   post;
   close;

   end; {with Table1}


{Send the Thank you Paper page with the link to the On-Line Article Review Page}
send ('<HTML><HEAD>');
SendTitle('On-Line Article Review System');

send ('</HEAD><BODY bgcolor=FFFFFF>');
send ('<center><H1>Edit a Review</H1></center>');

sendhr;
```

```
   send ('<CENTER><P>Thank you for your input,<B> '+ dReviewerID +'</B>.');
   send ('Your information has been added to the article review database.</P></CENTER>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Editrev.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Query another
Article"></CENTER></FORM>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Revpwd.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<INPUT TYPE="hidden" Name="ReviewerID" Value="'+ dReviewerID +'">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Return to the On-Line Article Review Options
Page"></CENTER></FORM>');

   send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
   send ('</BODY></HTML>');

   closeStdout;
   closeApp( application );      { don't leave form around }

   end  {flag is set}

   else begin    {flag **incorrect**}

   send ('<center><H2>On-line Article Review System</H2></center>');
   sendhr;
   send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
   send ('<P>Please ensure that you are authorized to access this information.');
   send ('<P>If you made an error, then please <A HREF="../review/Review30.htm">try to login
again.</A>');
   send ('<P>If you are not authorized to access this information, please note that a log is');
   send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
   send ('prosecution for your actions.<BR>');

   send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
   send ('Return to the <A HREF="../review/Review30.htm">On-line Article Review Log-on Page</A>');
   send ('<BR>');
   send ('</BODY></HTML>');
   closeStdout;
   closeApp( application );

        end;   {flag **incorrect**}

      end;   {with CGIEnvData1}

   end;  {FormCreate}

end.
```

APPENDIX D. MASTER REVIEW SUB-SYSTEM CODE

This appendix contains the HTML and Delphi code for the HTML documents and Delphi executables that comprise the Master Review Sub-system. The relationships of the programs are depicted in Figure D-1.

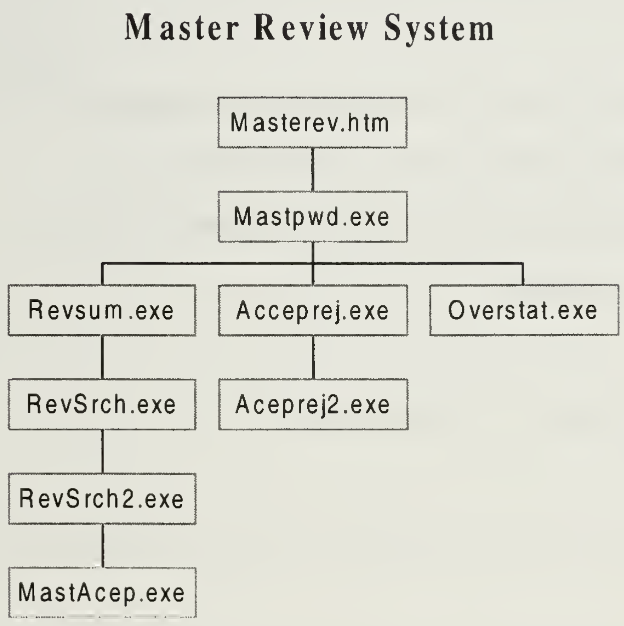


Figure D - 1.

The Master Review System log-on page: **Masterev.htm**

```
<HTML>

<HEAD>

<TITLE>Master Review System</TITLE>

</HEAD>

<BODY bgcolor=FFFFFF>

<center><H1>Master Review System</center></H1>

<P> This page is intended for the exclusive use by the Master Reviewer for
the Asilomar Conference on Signals, Systems, & Computers.

<FORM ACTION="../cgi-win/Mastpwd.exe " METHOD="POST">

<HR><CENTER>

<B>Master ID:    </B><INPUT NAME="masterid" Size="20" TYPE="text">

<B>Password:    </B><INPUT NAME="pwd" Size="20" TYPE="password">

<P><INPUT TYPE="submit" Value="Submit Password">  <INPUT TYPE="reset"'

VALUE="Clear Values"></CENTER>

<P><hr><P>

<IMG SRC="asil2b0.gif " align=left alt="Asilomar facility">

Go to the <A

HREF="http://XXX.XXX.XXX.XXX/submit/Index.htm">On-line Submission Page</A><BR>

Go to the <A

HREF="http://XXX.XXX.XXX.XXX/review/review30.htm">On-line Article Review Page</A><BR>

</BODY>

</HTML>
```

The Master Review System Main Menu page: **Mastpwd.exe**

```
unit Mast_pwd;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  Query1: TQuery;
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  Query1Password: TStringField;
  Table1Number: TIntegerField;
  Table1MasterID: TStringField;
  Table1Password: TStringField;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

 dMasterID: string;
 dPassword: string;
 dpwd: string;
 dflag: string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

 with CGIEnvData1 do begin

   webSiteINIFilename:=paramstr(1);
   application.onException:=cgiErrorHandler;
   application.processMessages;

   createStdout;
   sendPrologue;

   {Get fields from search page}
   dMasterID:= getsmallfield ('MasterID');
   dpwd:= getsmallfield ('pwd');
  dflag:= getsmallfield ('flag');

  send ('<HTML><HEAD>');
  SendTitle('Master Review System');      { Every page gets this }
  send ('</HEAD><BODY bgcolor=FFFFFF>');
  end;

{Retrieve Password of the given User}
with Query1 do begin

  close;
  sql.clear;
  sql.add('Select*FROM Master WHERE MasterID = ''' + dMasterID + '''');
  open;

  Table1.Open;
  Table1.First;
  dPassword := fieldByName('Password').Asstring;
  Table1.Close;
  end;


with cgiEnvData1 do begin

   {Determine whether password was correct}
   if (dPassword = dpwd) or (dflag = '1') then begin          {password was correct or flag is set}

   send('<center><H1>Master Review System</H1></center>');
   sendhr;
   send('<center><h3>Master Review Functions</h3></center>');

   {Summary of Reviews Button}
   send('<P><FORM ACTION="../cgi-win/Revsum.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<P><center><INPUT TYPE="submit" Value="Summary of Reviews"></center></FORM>');

   {Accept/Reject Articles Button}
   send('<P><FORM ACTION="../cgi-win/Acceprej.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<P><center><INPUT TYPE="submit" Value="Accept/Reject Articles"></center></FORM>');

   {Overall Submission Status Button}
   send('<P><FORM ACTION="../cgi-win/Overstat.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<P><center><INPUT TYPE="submit" Value="Overall Submission
Status"></center></FORM>');

   end

   else begin     {Password **incorrect**}

   send('<center><H1>Master Review System</H1></center>');
   sendhr;
   send('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
   send ('<P>Please ensure that you are authorized to access this information.');
```

```
    send ('<P>If you made an error, then please <A HREF="../review/masterev.htm">try to login
again.</A>');
    send ('<P>If you are not authorized to access this information, please note that a log is');
    send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
    send ('prosecution for your actions.<BR>');

    end;  {flag incorrect}

    send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
    send ('Go to the <A HREF="../review/Review30.htm">On-Line Article Review Page</A>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );

      end;   {with CGIEnvData1 do}
   end;   {FormCreate}
end.
```

The Summary of Reviews search options page: **Revsum.exe**

```
unit Rev_sum;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   DataSource1: TDataSource;
   Query1: TQuery;
   Table1: TTable;
   Table2: TTable;
   Table3: TTable;
   Table2Paper_Number: TIntegerField;
   Table2Title: TStringField;
   Table2Invited: TStringField;
   Table2Accepted: TStringField;
   Table2ContactAuthorNumber: TIntegerField;
   Table2Session: TStringField;
   Table2OrderInSession: TIntegerField;
   Table2PresentationTime: TStringField;
   Table2Keyword1: TStringField;
   Table2Keyword2: TStringField;
   Table2Keyword3: TStringField;
   Table2Abstract: TMemoField;
   Table2ContactOrder: TStringField;
   Table2FName2: TStringField;
   Table2LName2: TStringField;
   Table2Initial2: TStringField;
   Table2Institution2: TStringField;
   Table2Order2: TStringField;
   Table2FName3: TStringField;
   Table2LName3: TStringField;
   Table2Initial3: TStringField;
   Table2Institution3: TStringField;
   Table2Order3: TStringField;
   Table2FName4: TStringField;
   Table2LName4: TStringField;
   Table2Initial4: TStringField;
   Table2Institution4: TStringField;
   Table2Order4: TStringField;
   Table2FName5: TStringField;
   Table2LName5: TStringField;
   Table2Initial5: TStringField;
   Table2Institution5: TStringField;
   Table2Order5: TStringField;
   Table2FName6: TStringField;
   Table2LName6: TStringField;
   Table2Initial6: TStringField;
   Table2Institution6: TStringField;
   Table2Order6: TStringField;
   Table1RNumber: TIntegerField;
   Table1ReviewerID: TStringField;
   Table1Paper_Number: TStringField;
   Table1Title: TStringField;
   Table1Vote: TStringField;
   Table1Review: TMemoField;
   Table1Year: TStringField;
   Query1Name: TStringField;
   Query1Number: TIntegerField;
   Table3Number: TIntegerField;
   Table3Name: TStringField;
   DataSource2: TDataSource;
   Query2: TQuery;
   CGIDB2: TCGIDB;
   Table4: TTable;
   Query2ReviewerID: TStringField;
   Query2Number: TIntegerField;
   procedure FormCreate(Sender: TObject);

  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  Form1: TForm1;
  dflag: string;
  dReviewerID: string;
  dbOption: string;
  i:integer;           {loop control variable}


implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

    send ('<HTML><HEAD>');
    SendTitle('Master Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');

{Determine whether flag is valid }

dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

send('<center><H1>Summary of Reviews</H1></center>');
sendhr;

    send('<p>A summary of reviews in the database can be viewed for either all of the articles, ');
    send('or for reviews grouped by keyword.  Please select your reviewing option below.</p>');

  {Display ALL Article Reviews }
  send('<P><FORM ACTION="../cgi-win/RevSrch.exe" METHOD="POST">');
  send('<INPUT TYPE="hidden" Name="flag" Value="1">');
  send('<INPUT TYPE="hidden" Name="Type" Value="all">');

  sendhr;
  sendhdr ('3', 'Choose from a list of ALL Reviewed Articles?');
  send('<P><CENTER><INPUT TYPE="submit" Value="Show me a list of ALL
Reviews"></CENTER></FORM>');

  {Search by Keyword}
  send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/RevSrch.exe">');
  send('<INPUT TYPE="hidden" Name="flag" Value="1">');              {send flag for verification}
  send('<INPUT TYPE="hidden" Name="Type" Value="keyword">');

  sendhr;
  sendhdr ('3', 'Choose from a list of Reviewed Articles with these characteristics?');

        {Pull Keywords from KEYWORD Table and display on pull-down menu}
        with Query1 do begin

            close;
            sql.clear;
            sql.add('Select * FROM Keyword ');
            open;

            {Move records from query result to stringlist}
            Table3.open;
            Table3.first;

            send('<P><CENTER><B>Keyword: </B><SELECT NAME="Keyword">');
            send('<OPTION> Do not search on this field');

            while not Table3.EOF do begin

                dbOption:= Table3.fieldByName('Name').Asstring;
                send('<OPTION> ' +dbOption+ '');
                Table3.next;
            end;  {for all records in the query result}

            send('</SELECT></CENTER>');
        end;  {withQuery1}

    send('<P><CENTER><INPUT TYPE="submit" Value="Show me a list of Reviews like this"> ');
    send('<INPUT TYPE="reset" VALUE="Reset Article Fields"></CENTER></form>');


    {Search by ReviewerID}

    send('<P><FORM  METHOD="POST"  ACTION="../cgi-win/RevSrch.exe">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');               {send flag for verification}
    send('<INPUT TYPE="hidden" Name="Type" Value="reviewer">');

    sendhr;
    sendhdr ('3', 'Choose from a list of Reviewers?');


        {Pull ReviewerID's from Review Table and display on pull-down menu}
        with Query2 do begin

          close;
          sql.clear;
          sql.add('Select * FROM Review');
          open;

          {Move records from query result to stringlist}
          Table4.open;
          Table4.first;

          send('<P><CENTER><B>ReviewerID: </B><SELECT NAME="ReviewerID">');
          send('<OPTION> Do not search on this field');

          while not Table4.EOF do begin

               dbOption:= Table4.fieldByName('ReviewerID').Asstring;
               send('<OPTION> ' +dbOption+ '');
               Table4.next;
          end;  {for all records in the query result}

          send('</SELECT></CENTER>');

        end;  {withQuery2}

    send('<P><CENTER><INPUT TYPE="submit" Value="Show me a list of Reviews by this Reviewer"> ');
    send('<INPUT TYPE="reset" VALUE="Reset Article Field"></CENTER></form>');


    sendhr;
    send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
    send('<INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<P><CENTER><INPUT TYPE="submit" Value="Return to Master Review System Options
   Page"></CENTER></FORM>');

    send ('</BODY></HTML>');
    closeStdout;
```

```
 closeApp( application );

 end  {flag is set}

 else begin     {flag **incorrect**}

 send ('<center><H2>Master Review System</H2></center>');
 sendhr;
 send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
 send ('<P>Please ensure that you are authorized to access this information.');
 send ('<P>If you made an error, then please <A HREF="../review/admin20.htm">try to login again.</A>');
 send ('<P>If you are not authorized to access this information, please note that a log is');
 send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
 send ('prosecution for your actions.<BR>');

 send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
 send ('Return to the <A HREF="../review/masterev.htm">Master Review Log-on Page</A>');
 send ('<BR>');
 send ('</BODY></HTML>');
 closeStdout;
 closeApp( application );

    end;   {flag **incorrect**}

   end;   {with cgiEnvData1 do}

 end;   {FormCreate}

end.
```

The Summary of Reviews search results page: **Revsrch.exe**

```
unit Re_srch2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Table1: TTable;
    Query1: TQuery;
    Table2: TTable;
    DataSource2: TDataSource;
    Table2RNumber: TIntegerField;
    Table2ReviewerID: TStringField;
    Table2Paper_Number: TStringField;
    Table2Title: TStringField;
    Table2Vote: TStringField;
    Table2Review: TMemoField;
    CGIDB2: TCGIDB;
    Query2: TQuery;
    Query3: TQuery;
    Query4: TQuery;
    Query5: TQuery;
    Query6: TQuery;
    Query7: TQuery;
    Table3: TTable;
    Table4: TTable;
    Table5: TTable;
    Table6: TTable;
    Table7: TTable;
    DataSource3: TDataSource;
    DataSource4: TDataSource;
    DataSource5: TDataSource;
    DataSource6: TDataSource;
    DataSource7: TDataSource;
    CGIDB3: TCGIDB;
    CGIDB4: TCGIDB;
    CGIDB5: TCGIDB;
    CGIDB6: TCGIDB;
    CGIDB7: TCGIDB;
    CGIDB8: TCGIDB;
    Query8: TQuery;
    DataSource8: TDataSource;
    Table8: TTable;
    Table9: TTable;
    procedure FormCreate(Sender: TObject);

  private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;
 dflag: string;
 dbOption: string;
 i, count, count1, count2, count3, count4, count5, count6 :integer;     {loop control variable}
 temp:string;
 countstr: string;
 countstr1: string;
 countstr2: string;
 countstr3: string;
 countstr4: string;
 countstr5: string;
 countstr6: string;

 theAuthors : TStringList;
 theabstract : TStringList;
 dbContactOrderInt: integer;
 dbOrder2int: integer;
 dbOrder3int: integer;
 dbOrder4int: integer;
 dbOrder5int: integer;
 dbOrder6int: integer;

 dbANumber: string;
 dbcontactOrder: string;

 dbfname2: string;
 dblname2: string;
 dbinitial2: string;
 dbinstitution2: string;
 dborder2: string;

 dbfname3: string;
 dblname3: string;
 dbinitial3: string;
 dbinstitution3: string;
 dborder3: string;

 dbfname4: string;
 dblname4: string;
 dbinitial4: string;
 dbinstitution4: string;
 dborder4: string;

 dbfname5: string;
 dblname5: string;
 dbinitial5: string;
 dbinstitution5: string;
 dborder5: string;
 dbfname6: string;
 dblname6: string;
 dbinitial6: string;
 dbinstitution6: string;
 dborder6: string;

 dbFName : string;
 dbLName : string;
 dbHonorific : string;
 dbInitial: string;
 dbInstitution : string;

 dReviewerID:  string;
 dVote:  string;
 dRNumber:  string;

 dPaper_Number: string;
 dTitle: string;

 dReview: string;
 dtype: string;

 dkeyword: string;

 MySelector: integer;
 TC: TDataSet;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

 with CGIEnvData1 do begin

   webSiteINIFilename:=paramstr(1);
   application.onException:=cgiErrorHandler;
   application.processMessages;

   createStdout;
   sendPrologue;

   send ('<HTML><HEAD>');
   SendTitle('Master Review System');
   send ('</HEAD><BODY bgcolor=FFFFFF>');

   {Determine whether flag is valid}
   dflag:= getsmallfield ('flag');
   dPaper_Number:= getsmallfield ('Paper_Number');

   if (dflag = '1') then begin    {flag is set}

     send('<center><H1>Master Review System</H1></center>');
     sendhr;
```

```
with Query8 do          {Check the database for the author First and Last name}
begin
   close;
   sql.clear;
   sql.add('Select*FROM Reviews WHERE Paper_Number="'+dPaper_Number+'"');
   open;

  {Get the Paper Record from the SUBMISSION Table}
  with Table8 do begin

   {Move to proper record}
   open;
   first;
   while fieldbyName('Paper_Number').asstring <> dPaper_Number do
       next;

   {Retrieve record}
   dTitle:= fieldByName ('Title').asstring;
   dbANumber:= fieldByName ('ContactAuthorNumber').asstring;

   CGIDB8.memoToStringList (fieldbyname ('Abstract'), theabstract);

   dbcontactOrder:= fieldByName ('contactOrder').asstring;

   dborder2:= fieldByName ('order2').asstring;
   dbfname2:= fieldByName ('fname2').asstring;
   dblname2:= fieldByName ('lname2').asstring;
   dbinstitution2:= fieldByName ('institution2').asstring;

   dborder3:= fieldByName ('order3').asstring;
   dbfname3:= fieldByName ('fname3').asstring;
   dblname3:= fieldByName ('lname3').asstring;
   dbinstitution3:= fieldByName ('institution3').asstring;

   dborder4:= fieldByName ('order4').asstring;
   dbfname4:= fieldByName ('fname4').asstring;
   dblname4:= fieldByName ('lname4').asstring;
   dbinstitution4:= fieldByName ('institution4').asstring;

   dborder5:= fieldByName ('order5').asstring;
   dbfname5:= fieldByName ('fname5').asstring;
   dblname5:= fieldByName ('lname5').asstring;
   dbinstitution5:= fieldByName ('institution5').asstring;

   dborder6:= fieldByName ('order6').asstring;
   dbfname6:= fieldByName ('fname6').asstring;
   dblname6:= fieldByName ('lname6').asstring;
   dbinstitution6:= fieldByName ('institution6').asstring;
   close;
   end;  {with Table8}

  {Get Author Name information from the AUTHOR Table}
  with Table9 do begin
   {Move to proper record}
   open;
```

```
first;
while fieldbyName('ANumber').asstring <> dbANumber do
    next;

{Retrieve Author Name information}
dbFName := fieldByName('FName').AsString;
dbLName := fieldByName('LName').AsString;
dbInstitution := fieldByName('Institution').AsString;

{Determine Order of Authors}
theAuthors:= TStringList.create;

{Send the Authors}
send ('<B>Author(s): </B><BR>');

if dbContactOrder = '' then
  send (' <I>'+ dbFName +' '+ dbLName +', '+dbInstitution+'</I> ')
  else begin

  {Initialize the stringlist}
  for i:= 0 to 6 do
     theAuthors.add('not used');

     dbContactOrderInt:= strtoint(dbContactOrder);
     theAuthors[dbContactOrderInt]:= '' +dbFName+ ' ' +dbLName+ ', '+dbInstitution+'';

     if dbOrder2 <> '' then begin
       dbOrder2Int:= strtoint(dbOrder2);
       theAuthors[dbOrder2Int]:= ''+dbFName2+ ' ' +dbLName2+ ', '+dbInstitution2+'';
       end;

     if dbOrder3 <> '' then begin
       dbOrder3Int:= strtoint(dbOrder3);
       theAuthors[dbOrder3Int]:= ''+dbFName3+ ' ' +dbLName3+ ', '+dbInstitution3+'';
       end;

     if dbOrder4 <> '' then begin
       dbOrder4Int:= strtoint(dbOrder4);
       theAuthors[dbOrder4Int]:= ''+dbFName4+ ' ' +dbLName4+ ', '+dbInstitution4+'';
       end;

     if dbOrder5 <> '' then begin
       dbOrder5Int:= strtoint(dbOrder5);
       theAuthors[dbOrder5Int]:= ''+dbFName5+ ' ' +dbLName5+ ', '+dbInstitution5+'';
       end;

     if dbOrder6 <> '' then begin
       dbOrder6Int:= strtoint(dbOrder6);
       theAuthors[dbOrder6Int]:= ''+dbFName6+ ' ' +dbLName6+ ', '+dbInstitution6+'';
       end;

  send ('<I>' +theAuthors[1]+ '</I><BR>');

  for i:= 2 to 6 do
     if theAuthors[i] <> 'not used' then send ('<I>' +theAuthors[i]+ '</I><BR>');
```

```
          end; {else}

        close;
        end;  {with Table9}

        send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
        send ('<H3><B>Title:  </B>  " ' + dTitle + ' " </H3>');
        send ('<B>Paper Number:    </B>    " ' + dPaper_Number + ' " ');
        send ('<P><B>Abstract:    </B><P>');
        for i := 0 to theabstract.count - 1 do
        send( theabstract.strings[i] );
        theabstract.free;  {release the memory held by 'theabstract'}

        send ('<P>You may view the extended summary for this submission by selecting');
        send ('file '+dPaper_Number+' from the<A HREF="../uploads/"> following list<A
HREF="../uploads/"></A>.<P>');
        sendhr;
        end; {with Query8}

       with Query1 do begin

          close;
          sql.clear;
          sql.add('Select * FROM Reviews WHERE Paper_Number="'+dPaper_Number+'"');
          open;
          count:= Recordcount;
          countstr:= inttostr (count);
          dTitle:= fieldbyname('Title').asstring;
          send('<P><center>There are ' +countstr+ ' reviews for this paper in the database.<P>');
          CGIDB1.drawtable;   {display all titles}
          send ('</center>');
          send ('<BR>');
          end;  {with Query1}
          send(' ');

       sendhr;

       with Query2 do begin

          close;
          sql.clear;
          sql.add('Select * FROM Reviews');
          sql.add('Where Paper_Number = "'+dPaper_Number+'"');
          open;
          count1:= Recordcount;
          countstr1:= inttostr(count1);
          close;
          end;  {with Query2}

       with Query3 do begin

          close;
          sql.clear;
          sql.add ('Select * FROM Reviews');
```

```
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Strong Accept"))');
    open;
    count2:= Recordcount;
    countstr2:= inttostr(count2);
    close;
    end;  {with Query3}

with Query4 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Conditional Accept"))');
    open;
    count3:= Recordcount;
    countstr3:= inttostr(count3);
    close;
    end;  {with Query4}

with Query5 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Reject"))');
    open;
    count4:= Recordcount;
    countstr4:= inttostr(count4);
    close;
    end;  {with Query5}

with Query6 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Undecided"))');
    open;
    count5:= Recordcount;
    countstr5:= inttostr(count5);
    close;
    end;  {with Query6}

with Query7 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Not Qualified"))');
    open;
    count6:= Recordcount;
    countstr6:= inttostr(count6);
    close;
    end;  {with Query7}
```

```
        send('<CENTER><B><H2>Vote Summary</H2></B>');
        send(' ');

        {Build list of Votes}
        send('<TABLE BORDER>');
        send('<TR><TH># Votes</TH><TH>Strong Accept</TH><TH>Conditional Accept</TH>');
        send('<TH>Reject</TH><TH>Undecided</TH><TH>Not Qualified</TH></TR>');
        send('<TR><TD>'+countstr1+'</TD>');
        send('<TD>'+countstr2+'</TD>');
        send('<TD>'+countstr3+'</TD>');
        send('<TD>'+countstr4+'</TD>');
        send('<TD>'+countstr5+'</TD>');
        send('<TD>'+countstr6+'</TD></TR>');
        send('</TABLE></CENTER>');
        send('<BR>');


        send ('<CENTER><B><I>Please cast your vote for this particular article:</I>');
        send('<P><FORM METHOD="POST" ACTION="../cgi-win/MastAcep.exe" >');
        send('<INPUT TYPE="hidden" Name="flag" Value="1">');
        send ('<INPUT TYPE="hidden" Name="Title" Value="'+ dTitle +'">');
        send ('<INPUT TYPE="hidden" Name="Paper_Number" Value="'+ dPaper_Number +'">');
        send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Accept" > Accept');
        send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Reject" > Reject</B>');
        send(' ');
        send('<P><INPUT TYPE="submit" Value="Submit Review Info"></CENTER></FORM>');

      {HTML Footer}
      sendhr;
      send('<P><FORM ACTION="../cgi-win/Revsum.exe" METHOD="POST">');
      send('<INPUT TYPE="hidden" Name="flag" Value="1">');
      send('<P><CENTER><INPUT TYPE="submit" Value="View another
summary"></CENTER></FORM>');

      sendhr;
      send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
      send('<INPUT TYPE="hidden" Name="flag" Value="1">');
      send('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review Options
Page"></CENTER></FORM>');
      send ('</BODY></HTML>');
      closeStdout;
      closeApp( application );

     end {flag is set}

     else begin    {flag **incorrect**}

     send('<center><H1>Master Review System</H1></center>');
     sendhr;
     send('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
     send ('<P>Please ensure that you are authorized to access this information.');
     send ('<P>If you made an error, then please <A HREF="../review/review30.htm">try to login
again.</A>');
     send ('<P>If you are not authorized to access this information, please note that a log is');
     send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
```

```
    send ('prosecution for your actions.<BR>');

    send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
    send ('Return to the <A HREF="../review/masterev.htm">Master Review Log-on Page</A>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );

      end;    {flag **incorrect**}

    end;    {with cgiEnvData1 do}

  end;    {FormCreate}

end.
```

The Edit a Review page: **Revsrch2.exe**

```
unit Re_srch2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Table1: TTable;
    Query1: TQuery;
    Table2: TTable;
    DataSource2: TDataSource;
    Table2RNumber: TIntegerField;
    Table2ReviewerID: TStringField;
    Table2Paper_Number: TStringField;
    Table2Title: TStringField;
    Table2Vote: TStringField;
    Table2Review: TMemoField;
    CGIDB2: TCGIDB;
    Query2: TQuery;
    Query3: TQuery;
    Query4: TQuery;
    Query5: TQuery;
    Query6: TQuery;
    Query7: TQuery;
    Table3: TTable;
    Table4: TTable;
    Table5: TTable;
    Table6: TTable;
    Table7: TTable;
    DataSource3: TDataSource;
    DataSource4: TDataSource;
    DataSource5: TDataSource;
    DataSource6: TDataSource;
    DataSource7: TDataSource;
    CGIDB3: TCGIDB;
    CGIDB4: TCGIDB;
    CGIDB5: TCGIDB;
    CGIDB6: TCGIDB;
    CGIDB7: TCGIDB;
    CGIDB8: TCGIDB;
    Query8: TQuery;
    DataSource8: TDataSource;
    Table8: TTable;
    Table9: TTable;
    procedure FormCreate(Sender: TObject);

  private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;
 dflag: string;
 dbOption: string;
 i, count, count1, count2, count3, count4, count5, count6 :integer;     {loop control variable}
 temp:string;
 countstr: string;
 countstr1: string;
 countstr2: string;
 countstr3: string;
 countstr4: string;
 countstr5: string;
 countstr6: string;

 theAuthors : TStringList;
 theabstract : TStringList;
 dbContactOrderInt: integer;
 dbOrder2int: integer;
 dbOrder3int: integer;
 dbOrder4int: integer;
 dbOrder5int: integer;
 dbOrder6int: integer;

 dbANumber: string;
 dbcontactOrder: string;

 dbfname2: string;
 dblname2: string;
 dbinitial2: string;
 dbinstitution2: string;
 dborder2: string;

 dbfname3: string;
 dblname3: string;
 dbinitial3: string;
 dbinstitution3: string;
 dborder3: string;

 dbfname4: string;
 dblname4: string;
 dbinitial4: string;
 dbinstitution4: string;
 dborder4: string;

 dbfname5: string;
 dblname5: string;
 dbinitial5: string;
 dbinstitution5: string;
 dborder5: string;
 dbfname6: string;
 dblname6: string;
 dbinitial6: string;
 dbinstitution6: string;
 dborder6: string;

 dbFName : string;
 dbLName : string;
 dbHonorific : string;
 dbInitial: string;
 dbInstitution : string;

 dReviewerID:  string;
 dVote:  string;
 dRNumber:  string;

 dPaper_Number: string;
 dTitle: string;

 dReview: string;
 dtype: string;

 dkeyword: string;

 MySelector: integer;
 TC: TDataSet;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

    send ('<HTML><HEAD>');
    SendTitle('Master Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');

    {Determine whether flag is valid}
    dflag:= getsmallfield ('flag');
    dPaper_Number:= getsmallfield ('Paper_Number');

    if (dflag = '1') then begin    {flag is set}

      send('<center><H1>Master Review System</H1></center>');
      sendhr;
```

```
with Query8 do          {Check the database for the author First and Last name}
begin
   close;
   sql.clear;
   sql.add('Select * FROM Reviews WHERE Paper_Number="'+dPaper_Number+'"');
   open;

  {Get the Paper Record from the SUBMISSION Table}
  with Table8 do begin

   {Move to proper record}
   open;
   first;
   while fieldbyName('Paper_Number').asstring <> dPaper_Number do
       next;

   {Retrieve record}
   dTitle:= fieldByName ('Title').asstring;
   dbANumber:= fieldByName ('ContactAuthorNumber').asstring;

   CGIDB8.memoToStringList (fieldbyname ('Abstract'), theabstract);

   dbcontactOrder:= fieldByName ('contactOrder').asstring;

   dborder2:= fieldByName ('order2').asstring;
   dbfname2:= fieldByName ('fname2').asstring;
   dblname2:= fieldByName ('lname2').asstring;
   dbinstitution2:= fieldByName ('institution2').asstring;

   dborder3:= fieldByName ('order3').asstring;
   dbfname3:= fieldByName ('fname3').asstring;
   dblname3:= fieldByName ('lname3').asstring;
   dbinstitution3:= fieldByName ('institution3').asstring;

   dborder4:= fieldByName ('order4').asstring;
   dbfname4:= fieldByName ('fname4').asstring;
   dblname4:= fieldByName ('lname4').asstring;
   dbinstitution4:= fieldByName ('institution4').asstring;

   dborder5:= fieldByName ('order5').asstring;
   dbfname5:= fieldByName ('fname5').asstring;
   dblname5:= fieldByName ('lname5').asstring;
   dbinstitution5:= fieldByName ('institution5').asstring;

   dborder6:= fieldByName ('order6').asstring;
   dbfname6:= fieldByName ('fname6').asstring;
   dblname6:= fieldByName ('lname6').asstring;
   dbinstitution6:= fieldByName ('institution6').asstring;
   close;
   end; {with Table8}

  {Get Author Name information from the AUTHOR Table}
  with Table9 do begin

   {Move to proper record}
```

```
open;
first;
while fieldbyName('ANumber').asstring <> dbANumber do
    next;

{Retrieve Author Name information}
dbFName := fieldByName('FName').AsString;
dbLName := fieldByName('LName').AsString;
dbInstitution := fieldByName('Institution').AsString;

{Determine Order of Authors}
theAuthors:= TStringList.create;

{Send the Authors}
send ('<B>Author(s): </B><BR>');

if dbContactOrder = '' then
  send (' <I>'+ dbFName +' '+ dbLName +', '+dbInstitution+'</I> ')
  else begin

  {Initialize the stringlist}
  for i:= 0 to 6 do
     theAuthors.add('not used');

     dbContactOrderInt:= strtoint(dbContactOrder);
     theAuthors[dbContactOrderInt]:= '' +dbFName+ ' ' +dbLName+ ', '+dbInstitution+'';

     if dbOrder2 <> '' then begin
       dbOrder2Int:= strtoint(dbOrder2);
       theAuthors[dbOrder2Int]:= ''+dbFName2+ ' ' +dbLName2+ ', '+dbInstitution2+'';
       end;

     if dbOrder3 <> '' then begin
       dbOrder3Int:= strtoint(dbOrder3);
       theAuthors[dbOrder3Int]:= ''+dbFName3+ ' ' +dbLName3+ ', '+dbInstitution3+'';
       end;

     if dbOrder4 <> '' then begin
       dbOrder4Int:= strtoint(dbOrder4);
       theAuthors[dbOrder4Int]:= ''+dbFName4+ ' ' +dbLName4+ ', '+dbInstitution4+'';
       end;

     if dbOrder5 <> '' then begin
       dbOrder5Int:= strtoint(dbOrder5);
       theAuthors[dbOrder5Int]:= ''+dbFName5+ ' ' +dbLName5+ ', '+dbInstitution5+'';
       end;

     if dbOrder6 <> '' then begin
       dbOrder6Int:= strtoint(dbOrder6);
       theAuthors[dbOrder6Int]:= ''+dbFName6+ ' ' +dbLName6+ ', '+dbInstitution6+'';
       end;

  send ('<I>' +theAuthors[1]+ '</I><BR>');

  for i:= 2 to 6 do
```

```
        if theAuthors[i] <> 'not used' then send ('<I>' +theAuthors[i]+ '</I><BR>');

        end; {else}

     close;
     end;  {with Table9}

     send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
     send ('<H3><B>Title:  </B>  " ' + dTitle + ' " </H3>');
     send ('<B>Paper Number:    </B>   " ' + dPaper_Number + ' " ');
     send ('<P><B>Abstract:    </B><P>');
     for i := 0 to theabstract.count - 1 do
     send( theabstract.strings[i] );
     theabstract.free;      {release the memory held by 'theabstract'}

     send ('<P>You may view the extended summary for this submission by selecting');
     send ('file '+dPaper_Number+' from the<A HREF="../uploads/"> following list<A
HREF="../uploads/"></A>.<P>');
     sendhr;
     end; {with Query8}

    with Query1 do begin

      close;
      sql.clear;
      sql.add ('Select * FROM Reviews WHERE Paper_Number="'+dPaper_Number+'"');
      open;
      count:= Recordcount;
      countstr:= inttostr (count);
      dTitle:= fieldbyname('Title').asstring;
      send ('<P><center>There are ' +countstr+ ' reviews for this paper in the database.<P>');
      CGIDB1.drawtable;  {display all titles}
      send ('</center>');
      send ('<BR>');
      end;  {with Query1}
      send(' ');

    sendhr;

    with Query2 do begin

      close;
      sql.clear;
      sql.add ('Select * FROM Reviews');
      sql.add ('Where Paper_Number = "'+dPaper_Number+'"');
      open;
      count1:= Recordcount;
      countstr1:= inttostr(count1);
      close;
      end;  {with Query2}

    with Query3 do begin

      close;
      sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Strong Accept"))');
    open;
    count2:= Recordcount;
    countstr2:= inttostr(count2);
    close;
    end;  {with Query3}

with Query4 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Conditional Accept"))');
    open;
    count3:= Recordcount;
    countstr3:= inttostr(count3);
    close;
    end;  {with Query4}

with Query5 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Reject"))');
    open;
    count4:= Recordcount;
    countstr4:= inttostr(count4);
    close;
    end;  {with Query5}

with Query6 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Undecided"))');
    open;
    count5:= Recordcount;
    countstr5:= inttostr(count5);
    close;
    end;  {with Query6}

with Query7 do begin

    close;
    sql.clear;
    sql.add ('Select * FROM Reviews');
    sql.add ('Where ((Paper_Number = "'+dPaper_Number+'")AND(Vote="Not Qualified"))');
    open;
    count6:= Recordcount;
    countstr6:= inttostr(count6);
    close;
    end;  {with Query7}
```

```
      send ('<CENTER><B><H2>Vote Summary</H2></B>');
      send (' ');

      {Build list of Votes}
      send ('<TABLE BORDER>');
      send ('<TR><TH># Votes</TH><TH>Strong Accept</TH><TH>Conditional Accept</TH>');
      send ('<TH>Reject</TH><TH>Undecided</TH><TH>Not Qualified</TH></TR>');
      send ('<TR><TD>'+countstr1+'</TD>');
      send ('<TD>'+countstr2+'</TD>');
      send ('<TD>'+countstr3+'</TD>');
      send ('<TD>'+countstr4+'</TD>');
      send ('<TD>'+countstr5+'</TD>');
      send ('<TD>'+countstr6+'</TD></TR>');
      send ('</TABLE></CENTER>');
      send ('<BR>');


      send ('<CENTER><B><I>Please cast your vote for this particular article:</I>');
      send ('<P><FORM METHOD="POST" ACTION="../cgi-win/MastAcep.exe" >');
      send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
      send ('<INPUT TYPE="hidden" Name="Title" Value="'+ dTitle +'">');
      send ('<INPUT TYPE="hidden" Name="Paper_Number" Value="'+ dPaper_Number +'">');
      send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Accept" > Accept');
      send ('<INPUT TYPE="radio" NAME="Vote" VALUE="Reject" > Reject</B>');
      send (' ');
      send ('<P><INPUT TYPE="submit" Value="Submit Review Info"></CENTER></FORM>');

     {HTML Footer}
     sendhr;
     send ('<P><FORM ACTION="../cgi-win/Revsum.exe" METHOD="POST">');
     send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
     send ('<P><CENTER><INPUT TYPE="submit" Value="View another
summary"></CENTER></FORM>');

     sendhr;
     send ('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
     send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
     send ('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review Options
Page"></CENTER></FORM>');
     send ('</BODY></HTML>');
     closeStdout;
     closeApp( application );

    end   {flag is set}

    else begin     {flag **incorrect**}

    send ('<center><H1>Master Review System</H1></center>');
    sendhr;
    send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
    send ('<P>Please ensure that you are authorized to access this information.');
    send ('<P>If you made an error, then please <A HREF="../review/review30.htm">try to login
again.</A>');
    send ('<P>If you are not authorized to access this information, please note that a log is');
    send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
    send ('prosecution for your actions.<BR>');

    send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
    send ('Return to the <A HREF="../review/masterev.htm">Master Review Log-on Page</A>');
    send ('</BODY></HTML>');
    closeStdout;
    closeApp( application );

      end;    {flag **incorrect**}

    end;    {with cgiEnvData1 do}

  end;    {FormCreate}

end.
```

The Master Review Accept/Reject page: **MastAcep.exe**

```
unit Mas_acep;

{WHEN CALLED: This program updates Accept/Reject data in the Submissi database.

 INPUT: Review data from the RevSrch2 page.

 ACTION: places Review data into the Submissi Table.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    Table1: TTable;
    DataSource1: TDataSource;
    procedure FormCreate(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dflag: string;
  dReviewerID: string;
  dVote:  string;
  dbTitle:  string;
  dPaper_Number: string;


implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);
begin

    with CGIEnvData1 do begin

    {Standard Header}
    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
```

```
   sendPrologue;

{Determine whether flag is valid }
dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

   {Get data from FORM}
   dPaper_Number:= getsmallfield ('Paper_Number');
   dVote:= getsmallfield ('Vote');

   {Place Data into Submissi Table}
   with Table1 do begin

      {Move to proper record}
      open;
      first;
      while fieldbyName('Paper_Number').asstring <> dPaper_Number do
           next;

      if (dVote = 'Accept') then begin    {Accept Article}

      {Update record}
      edit;
      Setfields ([nil, nil, nil, 'Y']);
      post;
      close;

      end  {vote is Accept}

      else begin  {vote is Reject}

      {Update record}
      edit;
      Setfields ([nil, nil, nil, 'N']);
      post;
      close;

      end; {if}

      end; {with Table1}

   {Send the Thank you Paper page with the link to the On-Line Article Review Page}
   send ('<HTML><HEAD>');
   SendTitle('Master Review System');

   send ('</HEAD><BODY bgcolor=FFFFFF>');
   send ('<center><H1>Accept/Reject an Article<BR></H1></center>');

   sendhr;
   send ('<P>Thank you for your input,<B> Master Reviewer</B>. ');
   send ('Your information has been added to the Article Submission database.</p>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/RevSrch.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<P><CENTER><INPUT TYPE="submit" Value="View another
Summary"></CENTER></FORM>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review Options
Page"></CENTER></FORM>');

   send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
   send ('</BODY></HTML>');
   closeStdout;
   closeApp( application );

  end  {flag is set}

  else begin     {flag **incorrect**}

  send ('<center><H2>Master Review System</H2></center>');
  sendhr;
  send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
  send ('<P>Please ensure that you are authorized to access this information.');
  send ('<P>If you made an error, then please <A HREF="../review/masterev.htm">try to login
again.</A>');
  send ('<P>If you are not authorized to access this information, please note that a log is');
  send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
  send ('prosecution for your actions.<BR>');

  send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
  send ('Return to the <A HREF="../review/masterev.htm">Master Review Log-on Page</A>');
  send ('<BR>');
  send ('</BODY></HTML>');
  closeStdout;
  closeApp( application );

      end;    {flag **incorrect**}

    end;    {with cgiEnvData1 do}

  end;    {FormCreate}

 end.
```

The Master Review Accept/Reject page: **Acceprej.exe**

```
unit Acc_rej;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Query1: TQuery;
    Articles: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dbtitle, dbPaper_Number: string;

  dflag: string;                  {valid program call}

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

     webSiteINIFilename:=paramstr(1);
     application.onException:=cgiErrorHandler;
     application.processMessages;
     createStdout;
     sendPrologue;

     send ('<HTML><HEAD>');
     SendTitle('Master Review System');
     send ('</HEAD><BODY bgcolor=FFFFFF>');

{Determine whether flag is valid}
dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

send('<center><H2>Master Review System</H2></center>');
```

```
sendhr;

 {Build Form}
     send('<P><FORM ACTION="../cgi-win/Aceprej2.exe" METHOD="GET">');

    {Build Pull-Down menu for choices}
     send('There are four methods of accepting or rejecting Articles:<BR><ul>');
     send('<LI>Accept the selected articles only, do not modify any other Articles');
     send('<LI>Accept the selected articles AND Reject all others');
     send('<LI>Reject the selected articles only, do not modify any other Articles');
     send('<LI>Reject the selected articles AND Accept all others</ul>');

     send('<P><center><SELECT NAME="Action">');
     send('<OPTION> ACCEPT');
     send('<OPTION> ACCEPT_then_Reject');
     send('<OPTION> REJECT');
     send('<OPTION> REJECT_then_Accept');
     send('</SELECT></center><P>');

   {Build the check boxes of Titles}
   with Articles do begin
      open;
      first;
      While not Articles.EOF do begin
          dbtitle:= fieldbyname('Title').asstring;
          dbPaper_Number:= fieldbyname('Paper_Number').asstring;
          send ('<INPUT TYPE="checkbox" NAME="T" Value="' +dbPaper_Number+ '"> # '
+dbPaper_Number+ ' ');
          send ('' +dbtitle+ ' <BR>');
          next;
          end;  {while not EOF}
      close;
      end;  {with articles do}

   send('<P><CENTER><INPUT TYPE="submit" Value=" Accept / Reject these articles "><P>  ');
   send('<INPUT TYPE="reset" VALUE="Reset values"></CENTER></FORM>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review System Options
page"></CENTER></FORM>');
end  {flag is set}

else begin     {flag **incorrect**}

send ('<center><H2>Master Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/masterev.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');
send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('Return to the <A HREF="http://XXX.XXX.XXX.XXX/review/masterev.html">Master');
send ('Review System Log-on Page</A>');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

   end;  {flag **incorrect**}

  send ('</BODY></HTML>');
  closeStdout;
  closeApp( application );

      end;   {with CGIEnvData1}

   end;   {FormCreate}

end.
```

The Master Review Accept/Reject confirmation page: **Aceprej2.exe**

```
unit Acc_rej2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Articles: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  i, j, dTitle, MySelector, dbPaper_Number: integer;

  dAction : string;
  dTitlein: string;
  dString, AssignString: string;

  Titlelist : TStringList;

  function TitleToInt ( S: string) :integer;


implementation

{$R *.DFM}
{***************************************}
function TitleToInt ( S: string) :integer;

{This function is necessary because the string value of dTitle has several spaces in
 it and the strtoint function cannot handle the conversion properly}

var
   temp: string;     {temporary variable to build new string into}
   j: integer;      {Counting Variable to keep track of how many characters are in the number}

begin
     {set the counter to zero}
     j:= 0;

     {count the characters}
    repeat
        j:= j+1;
    until S[j+1] = ' ';

    {Copy the valid number characters over to temp}
    temp := Copy(S, 1, j);

    {Convert the temp string to an integer value}
    TitleToInt := strtoint (temp);

end; {NoSpaces}

{**************************************}

procedure TForm1.formcreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;
    createStdout;
    sendPrologue;

    send ('<HTML><HEAD>');
    SendTitle('Master Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');
    send('<center><H2>Master Review System</H2></center>');
    sendhr;

    {Get query string from Assign.exe}
    Titlelist:= TStringlist.create;    {list of titles to assign to the session}

    AssignString:= CGIQueryString^;

    j:= 0;    {The counter for the individual strings within the QueryString}

    for i := 7 to Length(AssignString) do  {The counter for going through the AssignString - eight is the
                                            first valuable character  (the beginning of the Session
Neumonic) }

       if AssignString[i] <> '&' then begin    {the character is valuable}
          dString[j] := AssignString[i];
          j:= j+1;
          end {if AssignString[i] <> '&' then}
       else begin   {The & signifies the end of the string}
          TitleList.add (dString);
          dString:= '                              ';    {Blank out the old values}
          j:=0;              {reset the string counter}
          i:= i+1;           {Skip over the text "T="}
          end; {else}
```

```
TitleList.add (dString);   {This is to put the last string into the string list.  It is
                           necessary because there is no & at the end.}

{Place the Session Neumonic into each Paper_Number passed}

   {The first item in the Title List is the Session Neumonic}
   dAction:= TitleList[0];

   if (dAction[1]='A') and (dAction[7] <> '_') then MySelector := 1;
   if (dAction[1]='A') and (dAction[7] = '_') then MySelector := 2;
   if (dAction[1]='R') and (dAction[7] <> '_') then MySelector := 3;
   if (dAction[1]='R') and (dAction[7] = '_') then MySelector := 4;


case MySelector of

1 : begin    {Accept only}

         {Send confirmation}
         sendhdr ('3', '<CENTER><P>The following Articles were ACCEPTED:<br>');

         {accept the articles with the selected Paper_Numbers}
         with Articles do begin

           open;  {Prepare the Table for use}
           for i:= 1 to TitleList.count-1 do begin   {The rest of the items are Paper_Numbers}

             {Get the next title number}
             dTitlein:= TitleList[i];

             {Convert the Title string to an integer value}
             dTitle:= TitleToInt (dTitlein);

             {Move to proper record}
             first;    {go back to the first record before every new search}
             while fieldbyname('Paper_Number').asinteger <> dTitle do
               next;

             {Insert the Session}
             edit;
             Setfields ([nil, nil, nil, 'Y']);  {The remaining fields are ignored}
             post;

             {Send title name}
             send('# ' +fieldbyname('Paper_Number').asstring+ ' ' +fieldbyname('Title').asstring+ '<BR>');

             end;  {for}

           close; { the table}

           end;  {with Articles}
          send('</center>');
          end; {Accept only}
```

```
2 : begin   {Accept and Reject}

          {Send confirmation}
          sendhdr ('3', '<CENTER><P>The following Articles were ACCEPTED:<br>');

      {accept the articles with the selected Paper_Numbers}
          with Articles do begin

            open;  {Prepare the Table for use}

            {Set all *Not Invited* Articles to "rejected"}
            while not EOF do begin

              if fieldbyname('Invited').asstring <> 'Y' then begin

                edit;
                Setfields ([nil, nil, nil, 'N']);  {The remaining fields are ignored}
                post;
                end; {If not invited}

              next;
              end; {While}

            for i:= 1 to TitleList.count-1 do begin  {The rest of the items are Paper_Numbers}

              {Get the next title number}
              dTitlein:= TitleList[i];

              {Convert the Title string to an integer value}
              dTitle:= TitleToInt (dTitlein);

              {Move to proper record}
              first;   {go back to the first record before every new search}
              while fieldbyname('Paper_Number').asinteger <> dTitle do
                next;

              {Insert the Session}
              edit;
              Setfields ([nil, nil, nil, 'Y']);  {The remaining fields are ignored}
              post;

              {Send title name}
              send('# ' +fieldbyname('Paper_Number').asstring+ ' ' +fieldbyname('Title').asstring+ '<BR>');

              end;  {for}

            close; { the table}

            end;  {with Articles}

          sendhdr('3', '<P>All OTHER Articles were Rejected!</center><BR><P>');

          end; {Accept and Reject}
```

```
3 : begin   {Reject only}

          {Send confirmation}
          sendhdr ('3', '<CENTER><P>The following Articles were REJECTED:<br>');

          {The rest of the items are Paper_Numbers}
          with Articles do begin

            open;   {Prepare the Table for use}
            for i:= 1 to TitleList.count-1 do begin

              {Get the next title number}
              dTitlein:= TitleList[i];

              {Convert the Title string to an integer value}
              dTitle:= TitleToInt (dTitlein);

              {Move to proper record}
              first;    {go back to the first record before every new search}
              while fieldbyname('Paper_Number').asinteger <> dTitle do
                 next;

              {Insert the Session}
              edit;
              Setfields ([nil, nil, nil, 'N']);  {The remaining fields are ignored}
              post;

              {Send title name}
              send('# ' +fieldbyname('Paper_Number').asstring+ ' ' +fieldbyname('Title').asstring+ '<BR>');

              end;  {for}

            close; { the table}
            send('</center>');

            end;  {with Articles}

          end; {Reject only}

4: begin  {'Reject and accept'}

          {Send confirmation}
          sendhdr ('3', '<CENTER><P>The following Articles were REJECTED:<br>');

          {accept the articles with the selected Paper_Numbers}
          with Articles do begin

            open;   {Prepare the Table for use}

            {Set all *Not Invited* Articles to "accepted"}
            while not EOF do begin

              if fieldbyname ('Invited').asstring <> 'Y' then begin
```

```
                edit;
                Setfields ([nil, nil, nil, 'Y']);  {The remaining fields are ignored}
                post;
                end; {If not invited}

              next;
              end; {While}


             for i:= 1 to TitleList.count-1 do begin   {The rest of the items are Paper_Numbers}

               {Get the next title number}
               dTitlein:= TitleList[i];

               {Convert the Title string to an integer value}
               dTitle:= TitleToInt (dTitlein);

               {Move to proper record}
               first;    {go back to the first record before every new search}
               while fieldbyname('Paper_Number').asinteger <> dTitle do
                 next;

               {Insert the Session}
               edit;
               Setfields ([nil, nil, nil, 'N']);  {The remaining fields are ignored}
               post;

               {Send title name}
               send('# ' +fieldbyname('Paper_Number').asstring+ ' ' +fieldbyname('Title').asstring+ '<BR>');

               end;  {for}

             close; {  the table}

             end;  {with Articles}

           sendhdr('3', '<P>All OTHER Articles were Accepted!</center><BR><P>');

            end; {Reject and accept}
else begin {selector not working}

    send('Selector did not work<BR>');
    send('Action was ' +dAction+ '.');
    end; {else}

end; {case}

    {Send Footer}
    {Button to Accept/Reject more Articles}
    send('<P><FORM ACTION="../cgi-win/Acceprej.exe" METHOD="POST">');
    send('<CENTER><INPUT TYPE="hidden" Name="flag" Value="1">');
    send('<P><INPUT TYPE="submit" Value="Accept/Reject more Articles"></FORM><P>  ');
```

```
  {Button to move to Assign Articles to Sessions}
   send('<P><FORM ACTION="../cgi-win/Assign.exe" METHOD="POST">');
   send('<INPUT NAME="flag" TYPE="hidden" Value="1">');
   send('<P><INPUT TYPE="submit" Value="Now Assign the Articles to Sessions "
Size="30"></center></FORM>');

   sendhr;
   send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
   send('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review System Options
page"></CENTER></FORM>');
   send ('</BODY></HTML>');
   closeStdout;
   closeApp( application );

      end;    {with cgiEnvData1 do}

   end;    {FormCreate}

end.
```

The Master Review overall submission status page: **Overstat.exe**

```
unit Ovr_stat;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Query1: TQuery;
    Table1: TTable;
    Table1Paper_Number: TIntegerField;
    Table1Title: TStringField;
    Table1Invited: TStringField;
    Table1Accepted: TStringField;
    Table1ContactAuthorNumber: TIntegerField;
    Table1Session: TStringField;
    Table1OrderInSession: TStringField;
    Table1PresentationTime: TStringField;
    Table1Keyword1: TStringField;
    Table1Keyword2: TStringField;
    Table1Keyword3: TStringField;
    Table1Abstract: TMemoField;
    Table1ContactOrder: TStringField;
    Table1FName2: TStringField;
    Table1LName2: TStringField;
    Table1Initial2: TStringField;
    Table1Institution2: TStringField;
    Table1Order2: TStringField;
    Table1FName3: TStringField;
    Table1LName3: TStringField;
    Table1Initial3: TStringField;
    Table1Institution3: TStringField;
    Table1Order3: TStringField;
    Table1FName4: TStringField;
    Table1LName4: TStringField;
    Table1Initial4: TStringField;
    Table1Institution4: TStringField;
    Table1Order4: TStringField;
    Table1FName5: TStringField;
    Table1LName5: TStringField;
    Table1Initial5: TStringField;
    Table1Institution5: TStringField;
    Table1Order5: TStringField;
    Table1FName6: TStringField;
    Table1LName6: TStringField;
    Table1Initial6: TStringField;
    Table1Institution6: TStringField;
    Table1Order6: TStringField;
  DataSource2: TDataSource;
  Query2: TQuery;
  Table2: TTable;
  CGIDB2: TCGIDB;
  DataSource3: TDataSource;
  Query3: TQuery;
  Table3: TTable;
  CGIDB3: TCGIDB;
  CGIDB4: TCGIDB;
  DataSource4: TDataSource;
  Query4: TQuery;
  Table4: TTable;
  procedure FormCreate(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
 Form1: TForm1;
 dflag: string;
 dReviewerID: string;
 dbOption: string;
 i, count, count2, count3, count4 :integer;
 countstr, countstr2, countstr3, countstr4: string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

  with CGIEnvData1 do begin

    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

    send ('<HTML><HEAD>');
    SendTitle('Master Review System');
    send ('</HEAD><BODY bgcolor=FFFFFF>');

  {Determine whether flag is valid }
  dflag:= getsmallfield ('flag');

  if (dflag = '1') then begin    {flag is set}

  send('<center><H1>Overall Submission Status</H1></center>');
  sendhr;
```

```
send('<center>Here are the current figures on the overall submission and review process:<BR>');

with Query1 do begin

   close;
   sql.clear;
   sql.add ('Select * FROM Submissi');
   open;
   count:= Recordcount;
   countstr:= inttostr (count);
   end;  {with Query1}

with Query2 do begin

   close;
   sql.clear;
   sql.add ('Select DISTINCT (Title) FROM Reviews');
   open;
   count2:= Recordcount;
   countstr2:= inttostr (count2);
   end;  {with Query2}

with Query3 do begin

   close;
   sql.clear;
   sql.add ('Select Title FROM Submissi WHERE Accepted="Y"');
   open;
   count3:= Recordcount;
   countstr3:= inttostr (count3);
   end;  {with Query3}

with Query4 do begin

   close;
   sql.clear;
   sql.add ('Select Title FROM Submissi WHERE Invited="Y"');
   open;
   count4:= Recordcount;
   countstr4:= inttostr (count4);
   end;  {with Query4}

   send('<BR> ');

       {Build list of Votes}
       send('<CENTER><TABLE BORDER>');
       send('<TR><TH># Submissions</TH><TH># Reviewed</TH><TH># Accepted</TH>');
       send('<TH># Invited</TH></TR>');
       send('<TR><TD>'+countstr+'</TD>');
       send('<TD>'+countstr2+'</TD>');
       send('<TD>'+countstr3+'</TD>');
       send('<TD>'+countstr4+'</TD></TR>');
       send('</TABLE></CENTER>');
       send('<BR>');
```

```
sendhr;
send('<P><FORM ACTION="../cgi-win/Mastpwd.exe" METHOD="POST">');
send('<INPUT TYPE="hidden" Name="flag" Value="1">');
send('<P><CENTER><INPUT TYPE="submit" Value="Return to the Master Review Options
Page"></CENTER></FORM>');

send ('</BODY></HTML>');
closeStdout;
closeApp( application );

end  {flag is set}

else begin     {flag **incorrect**}

send ('<center><H2>Master Review System</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../review/Masterev.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');


send ('<P><hr><P><IMG SRC="../review/asil2b0.gif " align=left alt="Asilomar facility">');
send ('Return to the <A HREF="http://XXX.XXX.XXX.XXX/review/Masterev.html">Asilomar');
send ('Conference On-line Submission Page</A>');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

        end;   {flag **incorrect**}

     end;   {with cgiEnvData1 do}

  end;    {FormCreate}

end.  {Application}
```

APPENDIX E. MISCELLANEOUS CODE

This appendix contains the HTML and Delphi code for several HTML documents and Delphi executables that were added to the System Administration Sub-system. These programs simply provide improved functionality in maintaining the system.

The System Admin overall submission status page: **Overstat2.exe**

```
unit Ovr_sta2;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DBTables, DB, Cgidb, Cgi;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  DataSource1: TDataSource;
  Query1: TQuery;
  Table1: TTable;
  DataSource2: TDataSource;
  Query2: TQuery;
  Table2: TTable;
  CGIDB2: TCGIDB;
  DataSource3: TDataSource;
  Query3: TQuery;
  Table3: TTable;
  CGIDB3: TCGIDB;
  CGIDB4: TCGIDB;
  DataSource4: TDataSource;
  Query4: TQuery;
  Table4: TTable;
  procedure FormCreate(Sender: TObject);

 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;
 dflag: string;
 dReviewerID: string;
 dbOption: string;
 i, count, count2, count3, count4 :integer;
 countstr, countstr2, countstr3, countstr4: string;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

 with CGIEnvData1 do begin

   webSiteINIFilename:=paramstr(1);
  application.onException:=cgiErrorHandler;
  application.processMessages;

  createStdout;
  sendPrologue;

  send ('<HTML><HEAD>');
  SendTitle('System Administration');
  send ('</HEAD><BODY bgcolor=FFFFFF>');

{Determine whether flag is valid }
dflag:= getsmallfield ('flag');

if (dflag = '1') then begin    {flag is set}

send('<center><H1>Overall Submission Status</H1></center>');
sendhr;

send('<center>Here are the current figures on the overall submission and review process:<BR>');

with Query1 do begin

   close;
   sql.clear;
   sql.add ('Select * FROM Submissi');
   open;
   count:= Recordcount;
   countstr:= inttostr (count);
   end;  {with Query1}

with Query2 do begin

   close;
   sql.clear;
   sql.add ('Select DISTINCT (Title) FROM Reviews');
   open;
   count2:= Recordcount;
   countstr2:= inttostr (count2);
   end;  {with Query2}

with Query3 do begin

   close;
   sql.clear;
   sql.add ('Select Title FROM Submissi WHERE Accepted="Y"');
   open;
   count3:= Recordcount;
   countstr3:= inttostr (count3);
   end;  {with Query3}

with Query4 do begin

   close;
   sql.clear;
   sql.add ('Select Title FROM Submissi WHERE Invited="Y"');
```

```
      open;
      count4:= Recordcount;
      countstr4:= inttostr (count4);
      end;  {with Query4}

      send('<BR> ');

          {Build list of Votes}
          send ('<CENTER><TABLE BORDER>');
          send ('<TR><TH># Submissions</TH><TH># Reviewed</TH><TH># Accepted</TH>');
          send ('<TH># Invited</TH></TR>');
          send ('<TR><TD>'+countstr+'</TD>');
          send ('<TD>'+countstr2+'</TD>');
          send ('<TD>'+countstr3+'</TD>');
          send('<TD>'+countstr4+'</TD></TR>');
          send ('</TABLE></CENTER>');
          send ('<BR>');

sendhr;
send ('<P><FORM ACTION="../cgi-win/Adminpwd.exe" METHOD="POST">');
send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
send ('<P><CENTER><INPUT TYPE="submit" Value="Return to the System Admin Options
Page"></CENTER></FORM>');

send ('</BODY></HTML>');
closeStdout;
closeApp( application );

end  {flag is set}

else begin     {flag **incorrect**}

send ('<center><H2>System Administration</H2></center>');
sendhr;
send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
send ('<P>Please ensure that you are authorized to access this information.');
send ('<P>If you made an error, then please <A HREF="../admin/admin20.htm">try to login again.</A>');
send ('<P>If you are not authorized to access this information, please note that a log is');
send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
send ('prosecution for your actions.<BR>');

send ('<P><hr><P><IMG SRC="../admin/asil2b0.gif " align=left alt="Asilomar facility">');
send ('</BODY></HTML>');
closeStdout;
closeApp( application );

       end;     {flag **incorrect**}

     end;     {with cgiEnvData1 do}

   end;    {FormCreate}

end.  {Application}
```

The database purge page: **Timeout.exe**

```
unit Time_out;

 {WHEN CALLED: This program verifies that user wants to purge old data from the Asilomar
   database.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  DataSource1: TDataSource;
  procedure FormCreate(Sender: TObject);

 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
  Form1: TForm1;
  dflag: string;
  dAdminID: string;
  dUserID: string;
  Subcount,Revcount: integer;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

   with CGIEnvData1 do begin

   {Standard Header}
   webSiteINIFilename:=paramstr(1);
   application.onException:=cgiErrorHandler;
   application.processMessages;

   createStdout;
   sendPrologue;

 {Determine whether flag is valid }
 dflag:= getsmallfield ('flag');

 if (dflag = '1') then begin   {flag is set}

    {Send the confirmation page to System Administrator that records were deleted}
   send ('<HTML><HEAD>');
   SendTitle('System Administration');

   send ('</HEAD><BODY bgcolor=FFFFFF>');
   send ('<center><H1>Database Purge</H1></center>');
   sendhr;

   send ('<CENTER>Are you absolutely certain that you want to purge old records from the
database?<P>');
   send ('Both old submissions and reviews will be purged.  Press the button below to execute the
purge.</CENTER><P>');

   sendhr;
   send ('<P><FORM ACTION="../cgi-win/Timeout2.exe" METHOD="POST">');
   send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send ('<INPUT TYPE="hidden" Name="UserID" Value="'+dUserID+'">');
   send ('<P><CENTER><INPUT TYPE="submit" Value="Purge Old Records
Now!"></CENTER></FORM>');

   sendhr;
   send ('<P><FORM ACTION="../cgi-win/Adminpwd.exe" METHOD="POST">');
   send ('<INPUT TYPE="hidden" Name="flag" Value="1">');
   send ('<P><CENTER><INPUT TYPE="submit" Value="Return to the System Admin Main
Menu"></CENTER></FORM>');

   send ('<P><hr><P><IMG SRC="../admin/asil2b0.gif " align=left alt="Asilomar facility">');
   send ('</BODY></HTML>');
   closeStdout;
   closeApp( application );

  end  {flag is set}

  else begin     {flag **incorrect**}

  send ('<center><H2>System Administration</H2></center>');
  sendhr;
  send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
  send ('<P>Please ensure that you are authorized to access this information.');
  send ('<P>If you made an error, then please <A HREF="../admin/admin20.htm">try to login again.</A>');
  send ('<P>If you are not authorized to access this information, please note that a log is');
  send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
  send ('prosecution for your actions.<BR>');

  send ('<P><hr><P><IMG SRC="../admin/asil2b0.gif " align=left alt="Asilomar facility">');
  send ('Return to the <A HREF="../admin/admin20.htm">System Administration Log-on Page</A>');
  send ('<BR>');
  send ('</BODY></HTML>');
  closeStdout;
  closeApp( application );

     end;   {flag **incorrect**}
   end;   {with cgiEnvData1 do}
  end;   {FormCreate}
end.
```

The database purge confirmation page: **Timeout2.exe**

```
unit Time_ou2;

{WHEN CALLED: This program purges old data from the Asilomar database.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgidb, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
   CGIEnvData1: TCGIEnvData;
   CGIDB1: TCGIDB;
   Table1: TTable;
   DataSource1: TDataSource;
   Table2: TTable;
   procedure FormCreate(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
  Form1: TForm1;
  dflag: string;
  dAdminID: string;
  Subcount,Revcount: integer;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin

    with CGIEnvData1 do begin

    {Standard Header}
    webSiteINIFilename:=paramstr(1);
    application.onException:=cgiErrorHandler;
    application.processMessages;

    createStdout;
    sendPrologue;

 {Determine whether flag is valid }
 dflag:= getsmallfield ('flag');

 if (dflag = '1') then begin    {flag is set}

     {Get System Admin ID from FORM}
```

```
dAdminID:= getsmallfield ('UserID');

{Check Submissi Table for old data}
with Table1 do begin
   Subcount:=0;
   open;
   first;
   while not Table1.EOF do begin
     if StrToDateTime(FieldByName('Year').AsString)<(Now - 20) then begin {older than 43 days}
       Delete; {record}
       Subcount:=Subcount + 1;
     end
     else
       next;
     end; {while not EOF}
   close;
   end; {with Table1}


{Check Reviews Table for old data}
with Table2 do begin
   Revcount:=0;
   open;
   first;
   while not Table2.EOF do begin
     if StrToDateTime(FieldByName('Year').AsString)<(Now - 15) then begin {older than 43 days}
       Delete; {record}
       Revcount:=Revcount + 1;
     end
     else
       next;
     end; {while not EOF}
   close;
   end; {with Table2}

{Send the confirmation page to System Administrator that records were deleted}
send ('<HTML><HEAD>');
SendTitle('System Administration');

send ('</HEAD><BODY bgcolor=FFFFFF>');
send ('<center><H1>Database Purge</H1></center>');
sendhr;

if Revcount=0 then begin
send ('<CENTER>The are no old records in the database!<P>');
send ('No records were deleted.</CENTER><P>');
end
else begin
send ('<CENTER>The Records have been successfully purged from the database.<P>');
send ('Reviews.DB records deleted:  '+IntToStr(Revcount)+'.</CENTER><P>');
end;

sendhr;
send('<P><FORM ACTION="../cgi-win/Adminpwd.exe" METHOD="POST">');
send('<INPUT TYPE="hidden" Name="flag" Value="1">');
```

```
  send('<P><CENTER><INPUT TYPE="submit" Value="Return to the System Admin Options
Page"></CENTER></FORM>');

  send ('<P><hr><P><IMG SRC="../admin/asil2b0.gif " align=left alt="Asilomar facility">');
  send ('</BODY></HTML>');
  closeStdout;
  closeApp( application );

 end  {flag is set}

 else begin    {flag **incorrect**}

 send ('<center><H2>System Administration</H2></center>');
 sendhr;
 send ('<center><H2>Your password was <strong>not accepted!!</strong></H2></center>');
 send ('<P>Please ensure that you are authorized to access this information.');
 send ('<P>If you made an error, then please <A HREF="../admin/admin20.htm">try to login again.</A>');
 send ('<P>If you are not authorized to access this information, please note that a log is');
 send ('maintained that includes your URL.  You may be subject to civil and/or criminal ');
 send ('prosecution for your actions.<BR>');

 send ('<P><hr><P><IMG SRC="../admin/asil2b0.gif " align=left alt="Asilomar facility">');
 send ('Return to the <A HREF="../admin/admin20.htm">System Administration Log-on Page</A>');
 send ('<BR>');
 send ('</BODY></HTML>');
 closeStdout;
 closeApp( application );

    end;   {flag **incorrect**}

   end;   {with cgiEnvData1 do}

 end;   {FormCreate}

end.
```

LIST OF REFERENCES

1. Chalfant, Michael D. and Coats, Kevin M., *Design and Implementation of a World Wide Web Conference Information System*. Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1996.

2. Chapman, Davis, *Building Internet Applications with Delphi 2*. Indianapolis: Que, 1996.

3. Mudry, Robert J., *Serving the Web*. Scottsdale: Coriolis Group, Inc., 1995.

4. *Delphi for Windows*. Version 1. Computer Software. Borland International, Inc., Windows 3.1 or higher, disk, 1995.

5. http://super.sonic.net/ann/delphi/cgicomp/detail.html

6. http://website.ora.com/wsdocs/extending.html

7. http://www.wftpd.com/products.htm

8. http://software.ora.com

9. http://www.borland.com/delphi

10. http://www.law.cornell.edu/uscode/17/index.html

11. http://live.netscape.com/newsref/std/nsapi_vs_cgi.html

12. http://www.microsoft.com/msdn/sdk/platforms/doc/sdk/internet/src /isapimrg5.htm

13. http://salsa.walldata.com/center/sal.htm

References were accurate as of 11 March 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Monique P. Fargues 3
   Department of Electrical and Computer Engineering, Code EC/FA
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Professor Rex A. Buddenberg 1
   Department of Systems Management, Code SM/BU
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Professor Herschel H. Loomis, Jr. 1
   Department of Electrical and Computer Engineering, Code EC/LM
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Professor Reuben T. Harris 1
   Department of Systems Management, Code SM/HR
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. LCDR Todd M. Kinney 1
   R.R. 1 Box 212
   Peru, IN 46970